SAND REPORT

SAND2002-2174
Unlimited Release
Printed July 2002

A 3-D Vortex Code for Parachute Flow Predictions: Version 1.0

James H. Strickland, Greg. F. Homicz, Vicki L. Porter, 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.

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

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

SAND2002-2174
Unlimited Release
Printed July 2002

A 3-D Vortex Code for Parachute Flow Predictions: VIPAR Version 1.0

James H. Strickland, Greg F. Homicz, and Vicki L. Porter
Engineering Sciences Center
Sandia National Laboratories
P. O. Box 5800
Albuquerque, NM 87185-0835

Albert A. Gossler
Mechanical Engineering Department
Northern Arizona University
Flagstaff, AZ 86011

Abstract

This report describes a 3-D fluid mechanics code for predicting flow past bluff bodies whose surfaces can be assumed to be made up of shell elements that are simply connected. Version 1.0 of the VIPAR code (Vortex Inflation PARachute code) is described herein. This version contains several first order algorithms that we are in the process of replacing with higher order ones. These enhancements will appear in the next version of VIPAR. The present code contains a motion generator that can be used to produce a large class of rigid body motions. The present code has also been fully coupled to a structural dynamics code in which the geometry undergoes large time dependent deformations. Initial surface geometry is generated from triangular shell elements using a code such as Patran and is written into an ExodusII database file for subsequent input into VIPAR. Surface and wake variable information is output into two ExodusII files that can be post processed and viewed using software such as EnSight™.

(this page intentionally left blank)

Nomenclature

$[A]$ = transformation matrix

A_i = surface area of element i

$[A_i]$ = element

$[A_{ij}]$ = influence coefficient matrix

a = flat plate half width

$\vec{a}$ = vector of constants

$\vec{b}$ = vector of constants

$\vec{c}$ = vector of constants

cx = abbreviation for $\cos x$

$\hat{e}$ = unit normal in the direction of Γ

$\hat{e}_{\delta l}$ = unit vector in the direction of $\vec{\delta l}$

$\vec{F}_n$ = normal force vector

F^* = non-dimensional force $p\frac{\rho}{2g_c}L_0^2U_0^2$

g_c = $32.174\frac{\text{lb}_m}{\text{lb}_f}\frac{\text{ft}}{\text{sec}^2}$, $1.0\frac{\text{slug}}{\text{lb}_f}\frac{\text{ft}}{\text{sec}^2}$, $1.0\frac{\text{kg}}{\text{newton}}\frac{\text{m}}{\text{sec}^2}$, etc.

h = perpendicular from vortex filament to evaluation point

i = index

$\hat{i}$ = unit vector in x direction

j = index

$\hat{j}$ = unit vector in y direction

$\hat{k}$ = unit vector in z direction

L_0 = reference length

l^* = non-dimensional length $\frac{l}{L_0}$

$\{\tilde{m}_j\}$ = added mass vector per unit surface area

n = index

Nomenclature (continued)

N_S = number of sources

N_T = number of target points

N_v = number of vortons

$\hat{n}$ = surface normal unit vector

P = number of processors

p^* = non-dimensional pressure $p\frac{\rho}{2g_c}U_0^2$

q = core function parameter

$\vec{R}_c$ = translational displacement of body fixed coordinates

R_e = Reynold's number $\frac{L_0 U_0}{\nu}$

r = radius

$\vec{r}$ = position vector

$\vec{r}'$ = position vector at vorticity source

$\vec{r}_n$ = position vector on rigid body at point n

$\vec{r}_V$ = position vector of vorton center

$\vec{r}_{V\text{head}}$ = position vector of vorton head

$\vec{r}_{V\text{tail}}$ = position vector of vorton tail

sx = abbreviation for $\sin x$

t = time

t^* = non-dimensional time $t\,\frac{U_0}{L_0}$

$\vec{U}_c$ = translational velocity vector of body fixed coordinates

U_0 = reference velocity

U_∞ = velocity at infinity

u = velocity in x direction

u^* = non-dimensional velocity $\frac{u}{U_0}$

$\vec{u}$ = velocity vector

Nomenclature (continued)

u_s	=	slip velocity along surface
$\vec{u}_p$	=	velocity at point p
$\vec{u}_b$	=	body velocity
$\vec{u}_f$	=	fluid velocity
$\vec{u}_{wi}$	=	wake velocity induced at point i
u_θ	=	circumferential velocity
$\dot{u}_{bn}$	=	acceleration of body normal to its surface
V	=	volume
v	=	velocity in y direction
x	=	coordinate direction
y	=	coordinate direction
z	=	coordinate direction
α	=	angle associated with triangular elements
Δh	=	surface panel dimension
Δp	=	pressure jump across membrane
$\Delta\vec{r}_V$	=	change in position vector of vorton center
Δt	=	time step
Δt_f	=	fluid time step
Δt_s	=	structural time step
$\Delta\tau_s$	=	time since start of current fluid time step
$\Delta\phi$	=	jump in velocity potential
δA	=	element area
$\vec{\delta l}$	=	vorton length
ε_u	=	error in velocity u
ε_v	=	error in velocity v

η = parallel efficiency

Γ = circulation

Γ^* = non-dimensional circulation $\frac{\Gamma}{L_0 U_0}$

γ = vortex sheet strength

γ^* = non-dimensional vortex sheet strength $\frac{\gamma}{U_0}$

$\vec{\gamma}$ = vortex sheet strength vector

$\vec{\zeta}$ = unit vector in either x, y, or z direction

θ = angle

$\vec{\theta}_c$ = rotational displacement of body fixed coordinates

λ = wavelength

ν = kinematic viscosity

ν^* = non-dimensional kinematic viscosity $\frac{\nu}{L_0 U_0} = \frac{1}{R_e}$

ξ_Φ = core function for vector potential

ξ_u = core function for velocity

ρ = fluid density or $\left|\vec{r} - \vec{r}'\right| / \tilde{\sigma}$

σ = vortex tube core radius

$\tilde{\sigma}$ = vorton core radius

$\vec{\Upsilon}$ = vorton strength

$\vec{\Phi}$ = vector potential

ϕ = velocity potential

ϕ^* = non-dimensional velocity potential $\frac{\phi}{L_0 U_0}$

$\vec{\Omega}_c$ = rotational velocity vector of body fixed coordinates

$\vec{\omega}$ = vorticity vector

∇^* = non-dimensional gradient operator $L_0 \nabla$

Table of Contents

1 INTRODUCTION

1.1 Motivation

The motivation for the present work is to develop a 3-D fluid mechanics code for predicting the flow over inflating nuclear weapons parachutes. Ultimately, we hope to be able to predict the performance of such parachute systems so as to be able study the detailed effects of material aging and deployment conditions. This is an extremely challenging problem since the surface geometry is complex and changes continuously (sometimes in surprising ways) during the inflation process. One of the parachutes of interest is constructed from ribbons that are on the order of 2 inches across. The parachute itself is approximately 50 feet in diameter and produces an unsteady wake that is on the order of a quarter of a mile long by the time the parachute is fully inflated. Thus, important geometrical length scales vary by four orders of magnitude. If one attempts to resolve the boundary layers on the ribbon elements, length scales in the flow will vary by six to seven orders of magnitude.

1.2 Goals & Assumptions

The primary goal of this work is to produce a baseline code VIPAR_1.0 (Vortex Inflation PARachute code version 1.0) that will provide some near term capability while providing a platform for the development of enhanced solution algorithms. We have formalized the process of version control by placing the code into the Sierra environment, making use of the Sierra Code Management System (SCMS).

In this report we will discuss the underlying physics algorithms found in VIPAR_1.0. In general, VIPAR_1.0 contains the following assumptions:

- The flow is incompressible.
- The surface may be flexible or rigid.
- The surface elements are impervious.

During FY2002, VIPAR_1.0 was coupled to the structural dynamics code PRESTO. This work will be described in an upcoming SAND report. The possibility of adding a compressible flow capability to VIPAR is also being investigated. Initial work in this area is presented in Reference [1].

1.3 General Approach

This report describes a 3-D fluid mechanics code for predicting flow past bluff bodies whose surfaces are assumed to be made up of triangular shell elements that are simply connected. The flow is assumed to be incompressible. The present code contains a motion generator that can be used to produce a large class of rigid body motions. The present code has also been fully coupled to the PRESTO structural dynamics code in which the geometry undergoes large time dependent deformations. Initial surface geometry is generated from triangular shell elements using a code such as Patran and is written into an ExodusII database file for subsequent input into VIPAR. Surface and wake variable information is output into two ExodusII files that can be post processed (viewed and/or recorded) using software such as Mustafa or EnSight™ [2].

We have chosen to use a gridless vortex approach to model the flow field (grids are still required on the parachute surface) that is advantageous for several reasons. First of all, by definition, grid generation is not required. Secondly, the computational domain consists only of regions that contain vorticity (parachute surface and wake) and thirdly, there is little if any numerical diffusion. In general, vortex methods are numerically stable although the vortex stretching term in 3-D must be judiciously handled.

The computation at the beginning of each time step requires that the normal velocity boundary condition be satisfied. In order to satisfy this boundary condition, a vortex sheet is placed on the parachute surface that produces a normal velocity at the surface. The strengths are adjusted so that the sum of the normal velocity perturbations from the freestream, the wake vorticity, and the new surface vorticity equal the normal velocity of the surface itself. This requires that a set of linear equations be solved for the unknown surface vorticity sheet strengths. In VIPAR_1.0 we use a smoothed low order boundary element method that initially assumes piecewise constant potential on each triangular element. Once these constant potential values are obtained for each element they are smoothed so that a piecewise constant value of vorticity may be computed. In future versions of VIPAR, a higher order boundary element method developed by Gharakhani [3], [4] will be used.

Next, the surface vorticity sheet is split into two sheets (one on each side of the surface) so as to satisfy the tangential velocity boundary condition (no slip). In this operation, the sum of the two split sheet vorticity strengths are equal to the original sheet strength so that the normal boundary condition is still satisfied. In VIPAR_1.0 the sheets are then moved (diffused) away from the surface and converted to vortons with Gaussian cores. Since there are, in general, conflicting requirements for placement of the nascent vorton centers, the boundary layer will not be resolved except for low Reynolds numbers cases in which the surface has been finely meshed. Fortunately, resolution of the boundary layers is not critical in bluff body flows such as parachutes. Global parameters such as drag coefficients for parachutes have experimentally been shown to be essentially Reynolds number independent.

Once the vorticity has been shed from the surface, it is convected, stretched, and diffused in the flow. In VIPAR_1.0 there is no merging or splitting of vortex elements. The diffusion model used in VIPAR_1.0 contains a stretching term that enhances diffusion. For low Reynolds number flows, diffusion may cause the vorton cores to overlap excessively. For high Reynolds numbers, core growth is small. In future versions of VIPAR, we plan to implement merging and splitting algorithms to better manage vortex element strain and growth due to stretching and diffusion, respectively, as well as control the total number of vortons in the flow. We also plan to implement a Vortex Redistribution Method (VRM) for diffusion due to Shankar [5] that has been further developed by Gharakhani [6].

2 BODY MOTION GENERATOR

In order to prescribe the motion of a body in VIPAR_1.0, a subroutine BodyMotion was written that returns displacements and velocities for all nodes as a function of time and body motion type. A body motion type simply specifies the translational and rotational velocity time history of a coordinate system fixed in the body relative to some globally fixed coordinate system. In the BodyMotion subroutine, it is assumed that the body moves in a rigid fashion. Therefore, the displacements and velocities for each node can be computed by knowing their position vectors with respect to the body fixed coordinate system along with the translational and rotational velocity time history of the body fixed coordinate system.

A subroutine CoordMotion is called by BodyMotion and returns the translational and rotational position and velocity vectors for a given motion type. Different motion types will in general require a different set of inputs that specify amplitudes and timings. Each motion type is constructed by specifying the translational and rotational velocities as a function of time using step, ramp, and trigonometric functions. These functions can be integrated exactly so that displacement vectors can be computed precisely from the translational and rotational velocity time histories. It should be pointed out that once a body motion type is selected and the timing parameters chosen, input is required for amplitudes for each of the six degrees of freedom (three translational and three rotational velocities).

2.1 Available Body Motion Types

An infinite number of body motions (velocity time histories) can be constructed from the step, ramp, and trigonometric functions. At the time of this report, three body motion types are available in the subroutine CoordMotion. These motions utilize the basic step, ramp, and trigonometric functions.

2.1.1 Impulsive Motion

A simple impulsive motion that starts at t_o and ends at t_f. At t_f all motion ceases. Input as motion_num =1 in the subroutine CoordMotion. The translational and rotational velocity vectors associated with the body fixed coordinate system are given by:

$$\vec{U}_c(t) = \begin{cases} \vec{a} & \text{for} \quad t_0 < t < t_f \\ 0 & \text{for} \quad t < t_0 \text{ or } t > t_f \end{cases}, \quad \vec{\Omega}_c(t) = \begin{cases} \vec{b} & \text{for} \quad t_0 < t < t_f \\ 0 & \text{for} \quad t < t_0 \text{ or } t > t_f \end{cases}, \tag{1}$$

The translational and rotational displacement vectors associated with the body fixed coordinate system are given by:

$$\vec{R}_c(t) = \begin{cases} \vec{a}(t - t_0) & \text{for} \quad t_0 < t < t_f \\ 0 & \text{for} \quad t < t_0 \text{ or } t > t_f \end{cases},$$

$$\vec{\theta}_c(t) = \begin{cases} \vec{b}(t - t_0) & \text{for} \quad t_0 < t < t_f \\ 0 & \text{for} \quad t < t_0 \text{ or } t > t_f \end{cases}. \tag{2}$$

2.1.2 Ramp Motion

A ramp motion starting at t_o followed by a constant velocity phase beginning at t_1 and lasting until t_2 after which a negative ramp motion begins that brings the velocity to zero at t_f. At t_f all motion ceases. Input as motion_num =2 in the subroutine CoordMotion. The translational and rotational velocity and displacement vectors are given by:

$$\vec{U}_c = \begin{cases} \vec{a}(t-t_0) & \text{for} \quad t_0 < t < t_1 \\ \vec{a}(t_1-t_0) & \text{for} \quad t_1 < t < t_2 \\ \vec{a}(t_1-t_0)\left(\dfrac{t_f-t}{t_f-t_2}\right) & \text{for} \quad t_2 < t < t_f \\ 0 & \text{for} \quad t < t_0 \text{ or } t > t_f \end{cases}, \tag{3}$$

$$\vec{\Omega}_c = \begin{cases} \vec{b}(t-t_0) & \text{for} \quad t_0 < t < t_1 \\ \vec{b}(t_1-t_0) & \text{for} \quad t_1 < t < t_2 \\ \vec{b}(t_1-t_0)\left(\dfrac{t_f-t}{t_f-t_2}\right) & \text{for} \quad t_2 < t < t_f \\ 0 & \text{for} \quad t < t_0 \text{ or } t > t_f \end{cases}$$

$$\vec{R}_c = \begin{cases} \frac{1}{2}\vec{a}(t-t_0)^2 & \text{for} \quad t_0 < t < t_1 \\ \frac{1}{2}\vec{a}(t_1-t_0)(2t-t_0-t_1) & \text{for} \quad t_1 < t < t_2 \\ \frac{1}{2}\vec{a}(t_1-t_0)(2t-t_0-t_1-t_2+t_f) & \text{for} \quad t_2 < t < t_f \\ 0 & \text{for} \quad t < t_0 \text{ or } t > t_f \end{cases}, \tag{4}$$

$$\vec{\theta}_c = \begin{cases} \frac{1}{2}\vec{b}(t-t_0)^2 & \text{for} \quad t_0 < t < t_1 \\ \frac{1}{2}\vec{b}(t_1-t_0)(2t-t_0-t_1) & \text{for} \quad t_1 < t < t_2 \\ \frac{1}{2}\vec{b}(t_1-t_0)(2t-t_0-t_1-t_2+t_f) & \text{for} \quad t_2 < t < t_f \\ 0 & \text{for} \quad t < t_0 \text{ or } t > t_f \end{cases}.$$

2.1.3 Trigonometric Motion

A trigonometric motion (sin function with a phase angle) starting at t_o and ending at t_f. At t_f all motion ceases. Input as motion_num =3 in the subroutine CoordMotion. The translational and rotational velocity vectors associated with the body fixed coordinate system are given by:

$$\vec{U}_c(t) = \begin{cases} \vec{a}_1 \sin[2\pi(\vec{a}_2 \cdot \vec{\zeta})(t-t_0) + \vec{a}_3 \cdot \vec{\zeta}] & \text{for} \quad t_0 < t < t_f \\ 0 & \text{for} \quad t < t_0 \text{ or } t > t_f \end{cases},$$

$$\vec{\Omega}_c(t) = \begin{cases} \vec{b}_1 \sin[2\pi(\vec{b}_2 \cdot \vec{\zeta})(t-t_0) + \vec{b}_3 \cdot \vec{\zeta}] & \text{for} \quad t_0 < t < t_f \\ 0 & \text{for} \quad t < t_0 \text{ or } t > t_f \end{cases}. \tag{5}$$

Here, $\vec{\zeta}$ is equal to the unit vector in the coordinate direction associated with the particular component of $\vec{U}_c(t)$ or $\vec{\Omega}_c(t)$ that is being specified.

The translational and rotational displacement vectors associated with the body fixed coordinate system are given by:

$$\vec{R}_c(t) = \begin{cases} \dfrac{\vec{a}_1 \sin[\pi(\vec{a}_2 \cdot \vec{\zeta})(t-t_0)] \sin[\pi(\vec{a}_2 \cdot \vec{\zeta})(t-t_0) + \vec{a}_3 \cdot \vec{\zeta}]}{\pi(\vec{a}_2 \cdot \vec{\zeta})} & \text{for} \quad t_0 < t < t_f \\ 0 & \text{for} \quad t < t_0 \text{ or } t > t_f \end{cases},$$

$$\vec{\theta}_c(t) = \begin{cases} \dfrac{\vec{b}_1 \sin[\pi(\vec{b}_2 \cdot \vec{\zeta})(t-t_0)] \sin[\pi(\vec{b}_2 \cdot \vec{\zeta})(t-t_0) + \vec{b}_3 \cdot \vec{\zeta}]}{\pi(\vec{b}_2 \cdot \vec{\zeta})} & \text{for} \quad t_0 < t < t_f \\ 0 & \text{for} \quad t < t_0 \text{ or } t > t_f \end{cases} \tag{6}$$

2.2 Rigid Body Motion

The translational velocity vector of a nodal point on the body $\vec{U}(\vec{r}_n)$ located at point $\vec{r}_n$ can be expressed in terms of $\vec{U}_c(t)$ and $\vec{\Omega}_c(t)$ according to Reference [7] as:

$$\vec{U}(\vec{r}_n) = \vec{\Omega}_c(t) \times [\vec{r}_n - \vec{R}_c(t)] + \vec{U}_c(t), \tag{7}$$

where $\vec{R}_c(t)$ is again the position vector of the origin of the body fixed coordinate system and $\vec{\Omega}_c(t)$ is the angular velocity about the point $\vec{R}_c(t)$. It should be noted that $\vec{\Omega}_c(t)$ and $\vec{r}_n$ must be expressed in the fixed global reference frame just as are $\vec{U}_c(t)$ and $\vec{R}_c(t)$.

In order to express the nodal position $\vec{r}_n$ in the global reference frame, the following transformation is required:

$$\{r_n\}_{\text{Global}} = [A]\{r_n\}_{\text{Body Fixed}} + \vec{R}_c(t).$$
$$[A] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cx & -sx \\ 0 & sx & cx \end{bmatrix} \begin{bmatrix} cy & 0 & sy \\ 0 & 1 & 0 \\ -sy & 0 & cy \end{bmatrix} \begin{bmatrix} cz & -sz & 0 \\ sz & cz & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{8}$$

Here cx, sx, etc. are the cosine and sin of the angle of rotation about the x axis, etc. These angles follow the “right hand rule” and are always referenced to the global coordinate system. These angles represent instantaneous values for time t.

In order to express $\vec{\Omega}_c(t)$ in the body fixed reference frame in terms of the fixed global reference frame, the following transformation is required:

$$\{\Omega_c\}_{\text{Global}} = [A]\{\Omega_c\}_{\text{Body Fixed}}, \tag{9}$$

Combining Equations 7, 8, and 9 yields an expression for the nodal velocity $\vec{U}(\vec{r}_n)$ in the global coordinate system in terms of the angular velocity vector $\vec{\Omega}_c(t)$ and nodal position $\vec{r}_n$ referenced to the body fixed coordinate system:

$$\vec{U}(\vec{r}_n) = \{[A]\{\Omega_c\}_{\text{Body Fixed}}\} \times \{[A]\{r_n\}_{\text{Body Fixed}}\} + \vec{U}_c. \tag{10}$$

It should be noted that the values of $\vec{\Omega}_c(t)$ in Equations 1, 3, and 5 and values of $\vec{\theta}_c(t)$ in Equations 2, 4, and 6 are given in terms of the body fixed coordinate system whereas the velocity $\vec{U}_c(t)$ and $\vec{R}_c(t)$ are in terms of the global coordinate system.

3 CALCULATION OF SURFACE VORTICITY

3.1 Piecewise Constant Velocity-Potential Boundary Elements

Consider a boundary element that is constructed by placing two infinitely-thin constant velocity-potential patches next to each other as indicated in Figure 1. This arrangement is equivalent to placing concentrated line vortices along the edges of the patch. The circulation around the edges can be written as:

$$\Gamma = \oint u_\theta r d\theta = \int_{-\pi}^{\pi} \frac{1}{r}\frac{\partial \phi}{\partial \theta} r d\theta = \int_1^2 d\phi = \phi_2 - \phi_1, \tag{11}$$

where r and θ are the cylindrical coordinates (with respect to the plate edge) of the path indicated in Figure 1.

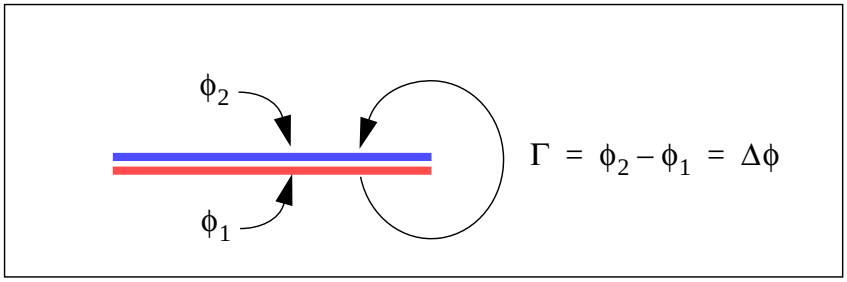


Figure 1. Constant Velocity-Potential Patches

3.1.1 Surface Discretization

In order to use boundary elements consisting of piecewise constant values of $\Delta\phi$ to obtain the surface vorticity flux, a triangular surface grid is first generated using a meshing program such as Patran™. An example for a circular disk is shown in Figure 2.

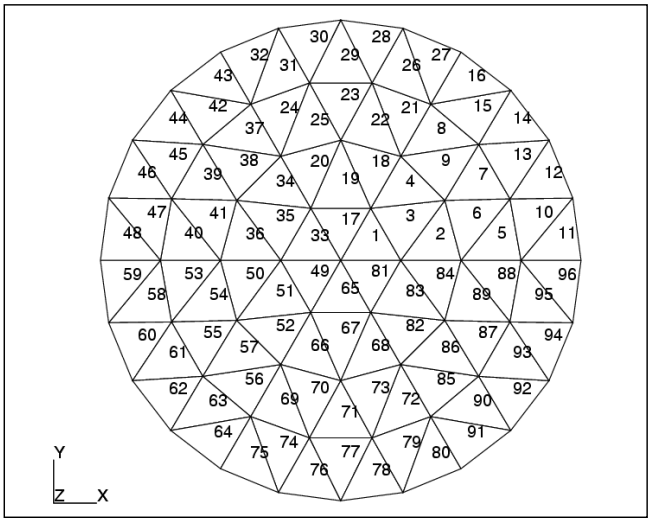


Figure 2. Example Surface Mesh for a Disk

As indicated previously, these patches give rise to vortex ring elements as shown in Figure 3. The positive direction of the surface normal $\hat{\boldsymbol{n}}$ and the assumed positive direction of Γ follow a "right-hand-rule" convention. In Figure 1, the normal to the surface patch would be downward according to this convention.

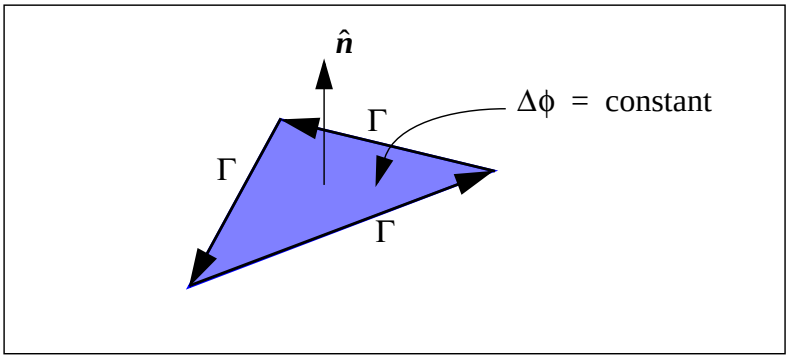


Figure 3. Triangular Vortex Ring Elements

3.1.2 Error Associated with the Vortex Loop Boundary Element

An estimate of the error that the vortex loop boundary elements will introduce on the velocity field near the surface may be made by considering the case shown in Figure 4. In this figure, a uniform vortex sheet with strength γ is discretized into an infinite series of discrete vortex filaments with strength Γ spaced along a surface at intervals of λ. These are formed from a set of rectangular vortex loops that have infinite extent in the z direction. The value of λ is equivalent to the element size in our vortex loop boundary element algorithm.

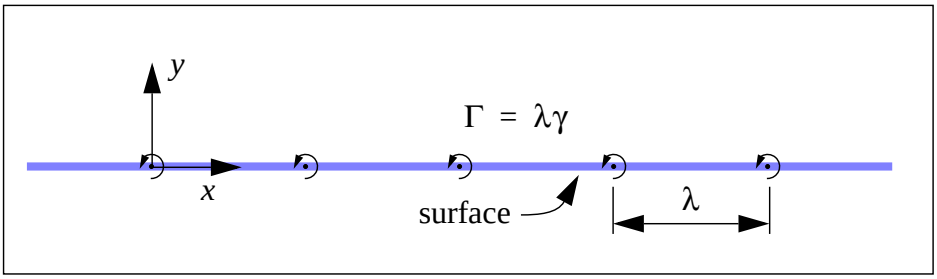


Figure 4. Discretization of a Uniform Vortex Sheet

From Lamb [8], the horizontal and vertical velocity components (u, v) for this system of vortices may be written as:

$$u(x,y) = \frac{\Gamma}{2\lambda}\frac{\sinh(2\pi y/\lambda)}{\cosh(2\pi y/\lambda)-\cos(2\pi x/\lambda)},$$
$$v(x,y) = \frac{\Gamma}{2\lambda}\frac{\sin(2\pi x/\lambda)}{\cosh(2\pi y/\lambda)-\cos(2\pi x/\lambda)}. \qquad (12)$$

In Figure 5, Equation 12 is plotted for two different distances away from the wall. The exact solution for a uniform vortex sheet would yield a value of $u = 0.5\gamma$ and $v = 0$.

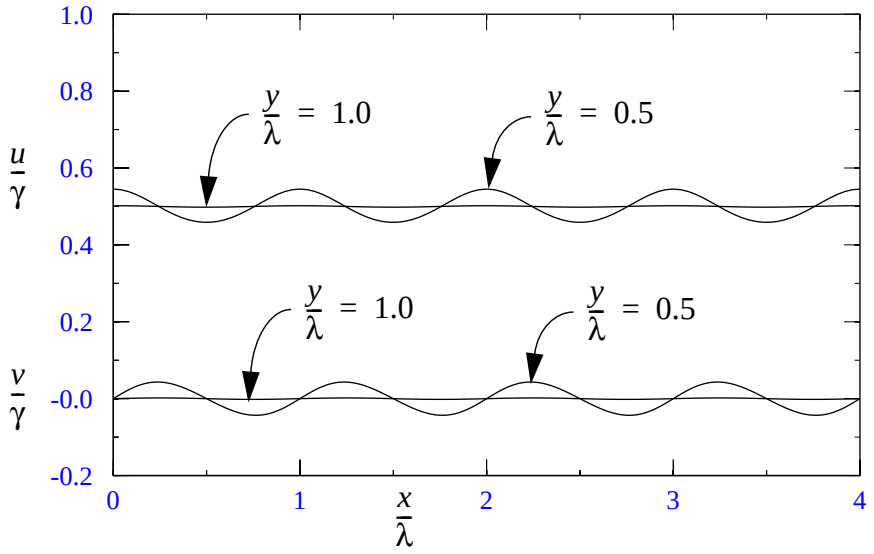


Figure 5. Velocities Near the Wall

We now define the error in the velocities as:

$$\varepsilon_u \equiv \left| \frac{max(u) - 0.5\gamma}{0.5\gamma} \right| \text{ and } \varepsilon_v \equiv \left| \frac{max(v)}{0.5\gamma} \right|. \qquad (13)$$

The maximum values of u and v occur at $x = 0$ and $x = \cos^{-1}[\text{sech}(2\pi y/\lambda)]\lambda/(2\pi)$ respectively. Using Equations 12 and 13, the error results are plotted in Figure 6. We note that the error decays at an exponential rate for $y/\lambda > 0.5$ and is on the order of 0.004 for $y/\lambda = 1.0$.

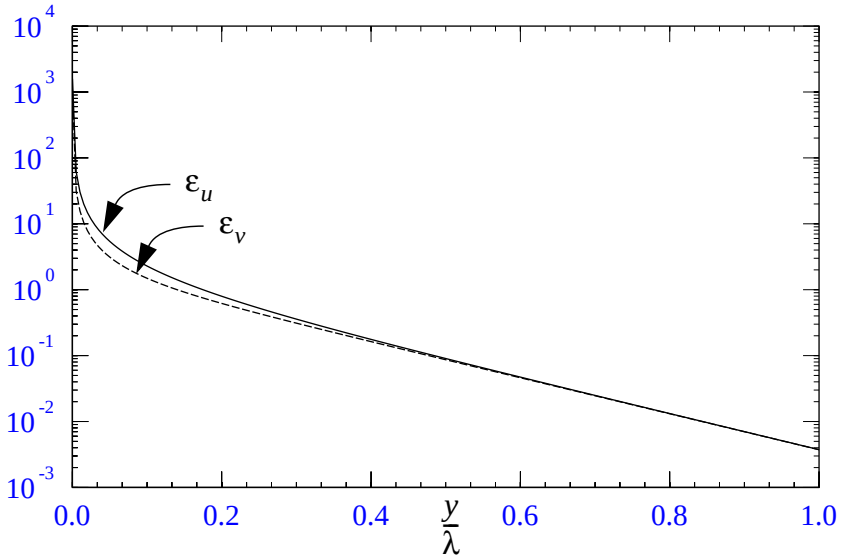


Figure 6. Velocity Errors in the Vortex Loop Boundary Element

3.2 Linear Equation Set

We first assume that the body is moving into an otherwise still fluid and that the body surface is impervious. The simplest method for determining the strengths Γ_j of the individual vortex rings is to require that the normal boundary condition be satisfied at a set of collocation points at the centroids of the triangular vortex rings. This gives rise to the following linear equation set:

$$\left[A_{ij}\right]\{\Gamma_j\} = \{\hat{n}_i \cdot (\vec{u}_{bi} - \vec{u}_{wi})\}. \tag{14}$$

The A_{ij} coefficients represent the normal velocity induced by the j^{th} source element (of unit strength) on the collocation point of element i. $\vec{u}_{bi}$ and $\vec{u}_{wi}$ are the body and wake velocities respectively at the collocation point on element i and $\hat{n}_i$ is the surface normal of element i.

Since the triangular ring element j is made up of three vortex filaments with strength Γ_j, then the velocity induced at a point p by this element can be readily constructed by use of Karamcheti's formula [9] for the induced velocity at a point p from a single vortex filament.

$$\vec{u}_p = \hat{e}\frac{\Gamma_j}{4\pi h}(\cos\theta_1 + \cos\theta_2). \tag{15}$$

Here, $\hat{e}$ is the unit vector in the direction of the element edge containing Γ_j, h is the perpendicular distance from the filament to the point p, and the angles θ_1, θ_2 are as indicated in Figure 7.

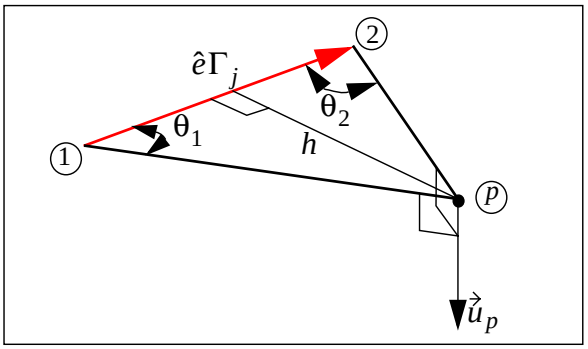


Figure 7. Perturbation Due to Vortex Filament

A convenient alternate form of Equation 15 is given by Strickland, Webster, and Nguyen [10]:

$$\vec{u}_p(\vec{a}, \vec{b}, \vec{c}) = \frac{\Gamma_j}{4\pi}\frac{\vec{c}\times\vec{a}}{|\vec{c}\times\vec{a}|^2}\left(\frac{\vec{a}\cdot\vec{b}}{b} - \frac{\vec{a}\cdot\vec{c}}{c}\right). \tag{16}$$

The vectors $(\vec{a}, \vec{b}, \vec{c})$ can be expressed in terms of the position vectors $(\vec{r}_1, \vec{r}_2, \vec{r}_p)$ of the points $(1, 2, p)$ by:

$$\begin{Bmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \end{Bmatrix} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 1 & -1 \\ 1 & 0 & -1 \end{bmatrix} \begin{Bmatrix} \vec{r}_1 \\ \vec{r}_2 \\ \vec{r}_p \end{Bmatrix}. \tag{17}$$

The coefficients A_{ij} may now be formulated for a triangular vortex ring with strength Γ_j and vertices (j_1, j_2, j_3) by the following:

$$A_{ij} = \hat{n}_i \cdot [\vec{u}_i(\vec{a}_1, \vec{b}_1, \vec{c}_1) + \vec{u}_i(\vec{a}_2, \vec{b}_2, \vec{c}_2) + \vec{u}_i(\vec{a}_3, \vec{b}_3, \vec{c}_3)] \ , \tag{18}$$

Here, $\vec{u}_i$ is the velocity at the collocation point on element i due to a unit source strength at j. The quantities $(\vec{a}_1, \vec{b}_1, \vec{c}_1)$, $(\vec{a}_2, \vec{b}_2, \vec{c}_2)$, and $(\vec{a}_3, \vec{b}_3, \vec{c}_3)$ are defined according to Equation 17 as:

$$\begin{bmatrix} \vec{a}_1 \\ \vec{b}_1 \\ \vec{c}_1 \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 1 & -1 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \vec{r}_{j_1} \\ \vec{r}_{j_2} \\ \vec{r}_{i_c} \end{bmatrix}, \quad \begin{bmatrix} \vec{a}_2 \\ \vec{b}_2 \\ \vec{c}_2 \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 1 & -1 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \vec{r}_{j_2} \\ \vec{r}_{j_3} \\ \vec{r}_{i_c} \end{bmatrix}, \quad \begin{bmatrix} \vec{a}_3 \\ \vec{b}_3 \\ \vec{c}_3 \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 1 & -1 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \vec{r}_{j_3} \\ \vec{r}_{j_1} \\ \vec{r}_{i_c} \end{bmatrix}. \tag{19}$$

where $\vec{r}_{i_c}$ is the position of the collocation point on the i^{th} element.

3.3 Pressure Jump Across Surface Elements

The pressure jump across the triangular patches of constant $\Delta\phi_i = \Gamma_i$ may be obtained by considering the impulse generated by the appearance of the vortex ring. According to Ashley and Landahl [11] the force normal to surface element i due to the appearance of a vortex ring of strength Γ_i is:

$$\vec{F}_{ni} = -\hat{n}_i \rho \frac{\partial}{\partial t}(A_i \Gamma_i), \tag{20}$$

where A_i is the area of element i and ρ is the fluid density. Therefore, the pressure jump across the surface (assuming that A_i is time invariant) is given by:

$$\Delta\vec{p}_i = -\hat{n}_i \rho \frac{\partial \Gamma_i}{\partial t} = -\hat{n}_i \rho \frac{\Gamma_i}{\Delta t}, \tag{21}$$

where Δt is the time step. It should be noted that Γ_i is actually the change in the vortex ring strength during the time step Δt since at the beginning of the time step the new ring has zero strength.

3.4 Smoothing Surface Vorticity

As noted in Section 3.1.2 on page 18, the error in the velocity induced by the vortex lattice can be quite large near the surface. Therefore, it is worthwhile to smooth the vorticity field for purposes of computing the surface induced velocity in that region. In order to gain some insight into the smoothing process, we first examine an impulsively started flat plate in which a smoothed vortex lattice solution is compared to the exact solution.

3.4.1 Impulsively Started 2-D Flat Plate

Consider a flat plate as shown in Figure 8 that is moved impulsively downward to a velocity equal to U_∞. The slip velocity at the plate surface is given by White [12] as:

$$\frac{u_s}{U_\infty} = \mp \frac{x/a}{(1 - x^2/a^2)^{1/2}}. \tag{22}$$

Here, the minus sign applies to the top surface and the plus sign applies to the bottom surface of the plate. From Equation 22 we can deduce the strength of the vorticity sheet on the plate that is the difference between the velocity on the bottom surface and velocity on the top surface.

$$\frac{\gamma}{U_\infty} = \frac{2x/a}{(1 - x^2/a^2)^{1/2}}. \tag{23}$$

The velocity potential jump $\Delta\phi$ as defined in Section 3.1 on page 17 is the negative of the integral of Equation 23 since $\gamma = \nabla_x(\phi_1 - \phi_2)$. At the plate edges $\Delta\phi = 0$ since we assume that there are no concentrated edge vortices. The velocity potential jump is then given by:

$$\frac{\Delta\phi}{aU_\infty} = (1 - x^2/a^2)^{1/2}. \tag{24}$$

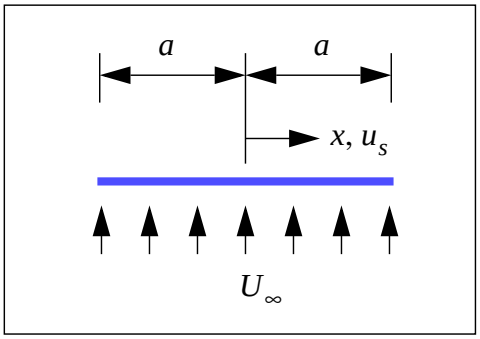


Figure 8. Impulsively Started Flat Plate

In order to construct a numerical solution, we subdivide the plate into n equal increments of length $\Delta h = 2a/n$. The center of the i^{th} increment or element x_i is given by:

$$\frac{x_i}{a} = \frac{2i - 1 - n}{n}, \qquad i = 1, n. \tag{25}$$

A constant value of $\Delta\phi_i$ is assumed for each element. Therefore, each element can be represented by a pair of vortices with equal but opposite strengths Γ_i centered at x_i and located a distance Δh apart. The normal velocity v_{ij} produced by the vortex pair of element i on the center of element j is given by:

$$\frac{v_{ij}}{U_\infty} = \frac{2a}{\pi\Delta h}\left[\frac{1}{1-4\left(\frac{x_i - x_j}{\Delta h}\right)^2}\right]\left(\frac{\Gamma_i}{aU_\infty}\right), \text{ where } i = 1, n \text{ and } j = 1, n. \tag{26}$$

Equation 26 may be rewritten in a simplified non-dimensional form as:

$$v_{ij}^* = A_{ij}\Gamma_i^*, \tag{27}$$

where

$$A_{ij} = \frac{2}{\pi\Delta h^*}\left[\frac{1}{1-4\left(\frac{x_i^* - x_j^*}{\Delta h^*}\right)^2}\right], \tag{28}$$

and

$$v_{ij}^* = \frac{v_{ij}}{U_\infty},\ \Gamma_i^* = \frac{\Gamma_i}{aU_\infty},\ x_i^* = \frac{x_i}{a},\ \Delta h^* = \frac{\Delta h}{a} \tag{29}$$

The linear equation set that is used to obtain Γ_i^* or $\Delta\phi_i^*$ is then given by:

$$[A_{ij}]\{\Gamma_i^*\} = \{1\}. \tag{30}$$

Numerical solution results for the potential function $\Delta\phi_i^* \equiv \Delta\phi/(aU_\infty) = \Gamma_i^*$ are plotted in Figure 9 for $n = 10$ along with the exact solution (Equation 24). The midpoint values of the piecewise constant solution are also indicated. This solution along with midpoint values of the numerical solutions for n equal to 20 and 40 are plotted in Figure 10. The error in $\Delta\phi_i^*$ at $x_i^* = 0$ ranges from about 5% for $n = 10$ to about 1% for $n = 40$.

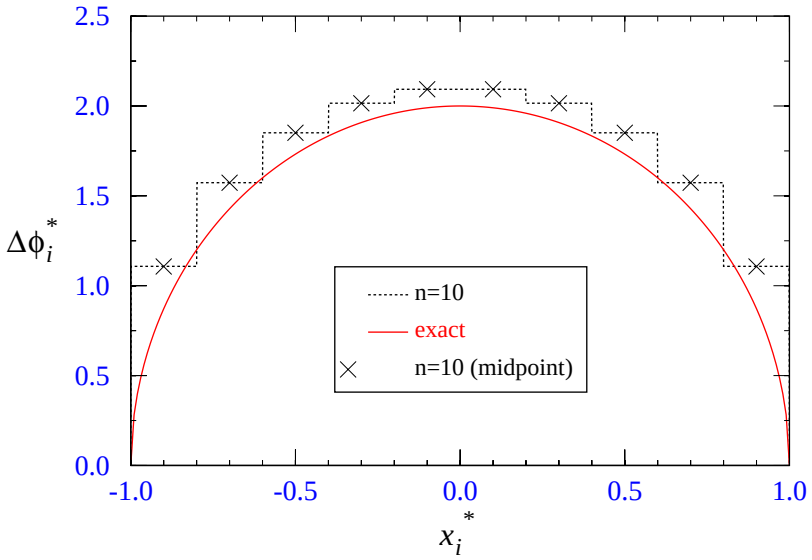


Figure 9. Piecewise Constant Potential Function for Impulsively Started Plate

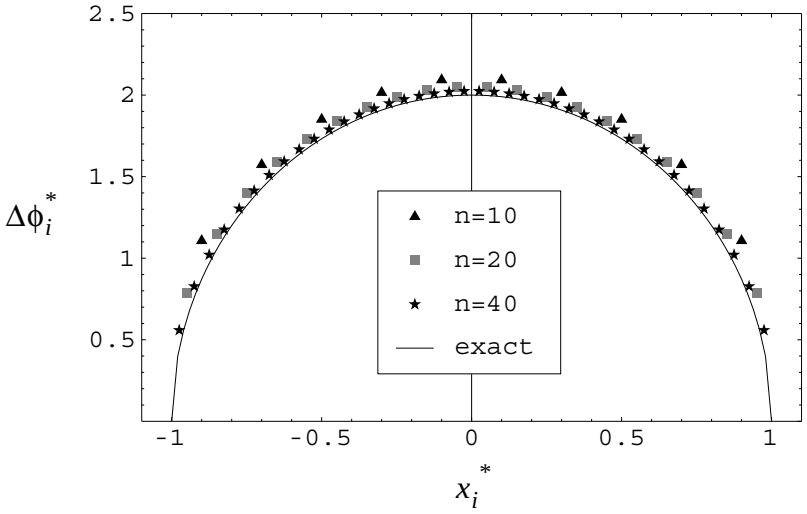


Figure 10. Potential Function for Impulsively Started Plate

In order to obtain piecewise constant vorticity, the velocity potential must be smoothed so as to be piecewise linear. One method of doing this that will later prove to be convenient in the three-dimensional case, is to average the potential at the jumps as indicated in Figure 11. A linear variation in potential is assumed between the averaged points. It should be noted that in the present scheme, the value of the smoothed potential at the plate edges in not equal to zero.

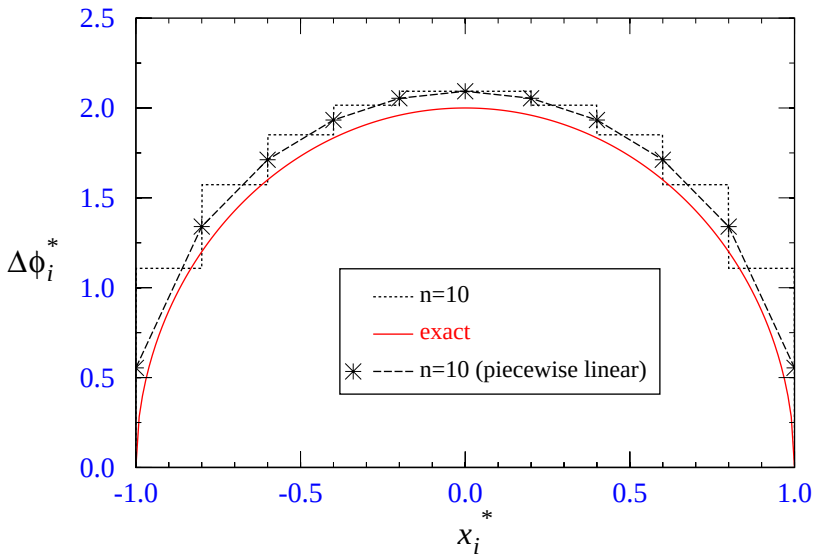


Figure 11. Smoothed Potential (piecewise linear)

The Piecewise constant values of vorticity obtained from the piecewise linear potential are shown in Figure 12 along with the exact solution (Equation 23). The results are seen to be quite good although it can be noted that near the edges that there are not enough data points to properly resolve the vortex sheet strength distribution.

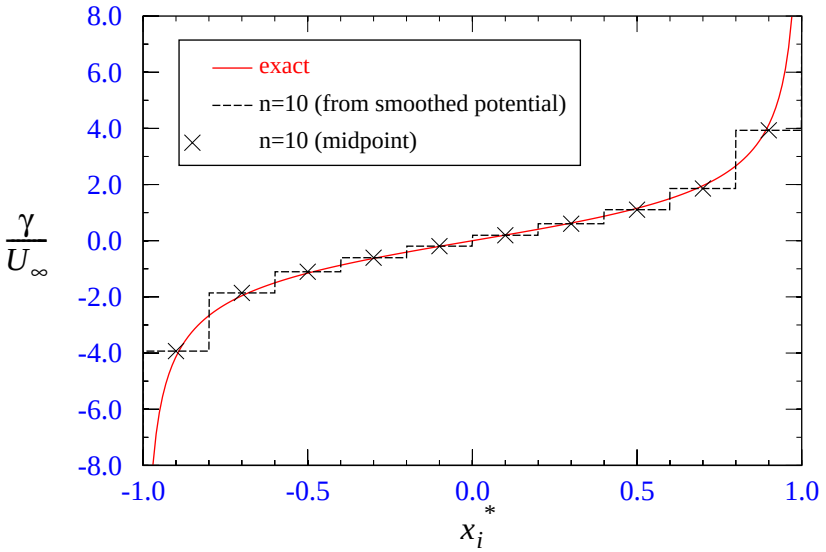


Figure 12. Piecewise Constant Vortex Sheet Strength from Smoothed Potential

In Figure 13, the exact solution and the solutions for the midpoint values of vorticity for *n* equal to 10, 20, and 40 are plotted. The results are seen to be very good.

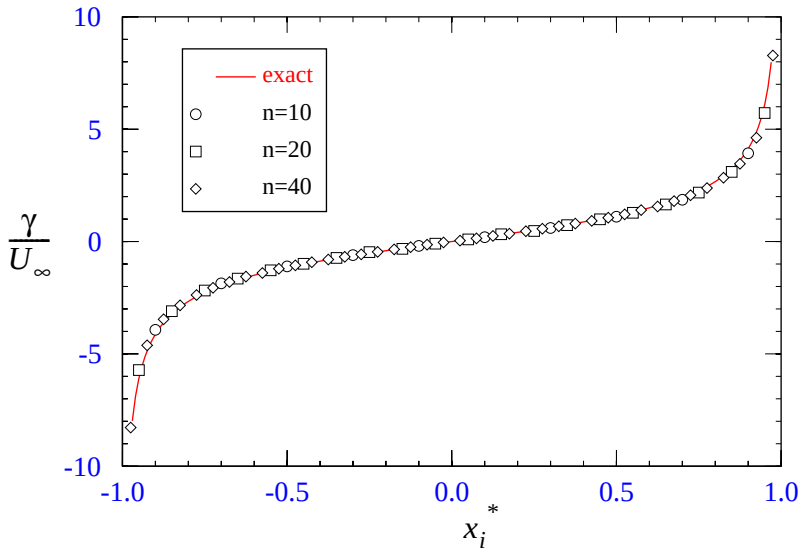


Figure 13. Vorticity Sheet for Impulsively Started Plate

There are several comments that need to be made about the vorticity distribution near the edges of the plate. First of all, we note that to be consistent, a concentrated vortex needs to be placed on each edge that has a strength equal to the smoothed value of $\Delta\phi_i$ at the plate edges as indicated in Figure 11. Since the vorticity in the exact case approaches infinity at the edges, a concentrated region of vorticity is indicated. If the value of the potential had been chosen to be equal to zero at the edge, no such concentrated vortex would be needed but the vorticity over the elements next to the edges would be considerably higher in magnitude. In essence, the latter approach uniformly distributes the concentrated edge vortex over the element at the edge.

It appears that the vorticity sheet could be further smoothed to yield a piecewise linear distribution by averaging the vorticity values at the element edges. The proper choice of the value of vorticity at the plate edges is, however, problematic. The crux of the problem is how best to distribute the concentrated vortex at the plate edge. Approaches include distributing the vorticity from the edge vortex uniformly over the edge element or using a triangular distribution over either the edge element or over the outer half of the edge element.

3.4.2 Smoothing in 3-D

For a three-dimensional body we assume that the vortex loop boundary element shown in Figure 3 is used. This produces a piecewise constant distribution of the potential $\Delta\phi$ as shown in Figure 14. As indicated by our numerical experiment with the 2-D flat plate, good results might be expected by averaging the potential jumps at the element boundaries. In three dimensions this means that we must obtain a suitable average at the nodes where several elements may be attached. A method that is consistent with the two-dimensional case is to

obtain a weighted average based on the “projected” included angles of the elements at a node. The term “projected,” in this case, means that the angles are projected onto a plane that is perpendicular to the node normal vector. The node normal vector is obtained from an average of element normals with included angles of the elements at the node used as weighting factors. For example, for the interior node 2 shown in Figure 14, the node normal and nodal value of potential are computed by using the following angle weighted averages:

$$\hat{n}_{node2} = \frac{\sum_{j=2}^{7} \alpha_{2j} \hat{n}_j}{\sum_{j=2}^{7} \alpha_{2j}} \tag{31}$$

$$\Delta\phi_{\text{node2}} = \frac{1}{2\pi} \sum_{j=2}^{7} (\hat{n}_j \cdot \hat{n}_{node2}) \alpha_{2j} \Delta\phi_j . \tag{32}$$

For the edge node 1, the nodal value is computed by using the same weighted average which includes the effect of the zero potential value off the body surface. Since the value of the potential along the edge is, in general, not zero then there should perhaps be a concentrated vortex whose strength varies linearly between edge nodal points. It is speculated that adding the edge vortices should improve the solution accuracy especially for poorly resolved geometries and during highly transient movements of the surface. Alternatively, we could set the potential jump equal to zero at the edge nodes since for incompressible flows there should be no pressure jump across the edge. For simplicity, in VIPAR_1.0 we have chosen to use this latter approach.

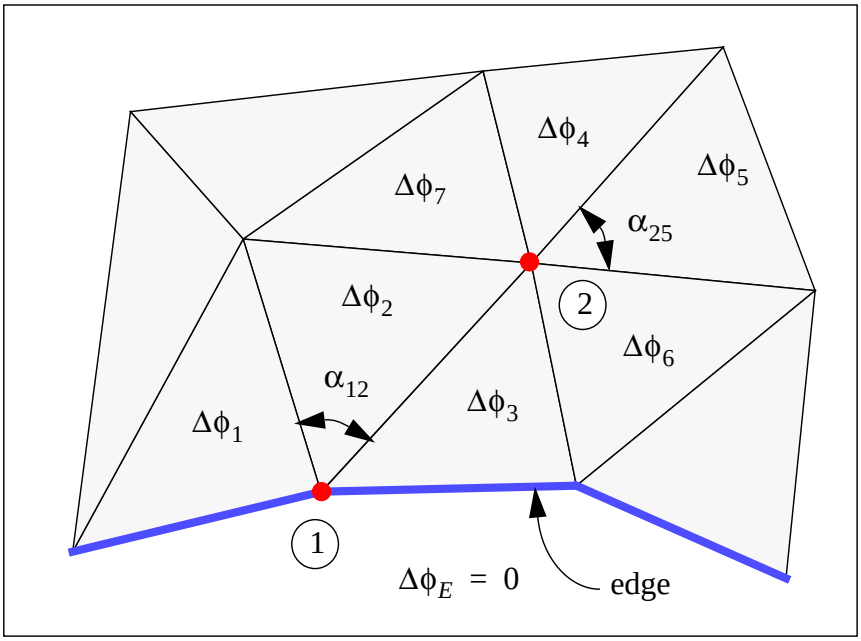


Figure 14. Averaged Nodal Potentials

Once the potential $\Delta\phi$ is smoothed onto the nodes, we assume a piecewise linear distribution of the potential over each triangular element. For each triangular element k, the potential $\Delta\phi_k$ may be written in the form:

$$\Delta\phi_k = a_0 + a_1 x_k + a_2 y_k , \tag{33}$$

where x_k and y_k lie in the plane of the triangular element. The coefficients (a_0, a_1, a_2) are obtained from the following linear equation set:

$$\begin{Bmatrix} \Delta\phi_{\text{node1}} \\ \Delta\phi_{\text{node2}} \\ \Delta\phi_{\text{node3}} \end{Bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix}, \tag{34}$$

where the $\Delta\phi_{\text{node}i}$ values are the averaged potentials at the three element nodes.

In order to obtain a piecewise constant distribution of vorticity on the elements, the vorticity vector for each element is obtained from:

$$\vec{\gamma}_k = -\hat{n}_k \times \nabla(\Delta\phi_k)_y = a_2 \hat{i}_k - a_1 \hat{j}_k , \tag{35}$$

where $\hat{i}_k$ and $\hat{j}_k$ are unit vectors that define the rectangular coordinate system in the plane of the element and $\hat{n}_k \equiv \hat{i}_k \times \hat{j}_k$. It can be noted that the vorticity vector is divergence free $(\nabla \cdot \vec{\gamma}_k = 0)$ on each element.

In order to obtain additional smoothing such that the surface vorticity is piecewise linear, the nodal vorticity vector is first computed. Again making reference to the example in Figure 14,

the nodal values are computed by using the following angle-weighted vector-summed average:

$$\vec{\gamma}_{\text{node2}} = \frac{\sum_{j=2}^{7} \alpha_{2j} \vec{\gamma}_j}{\sum_{j=2}^{7} \alpha_{2j}} . \tag{36}$$

For the exterior node 1, the nodal value is computed by using the same weighted average:

$$\vec{\gamma}_{\text{node1}} = \frac{\sum_{j=1}^{3} \alpha_{1j} \vec{\gamma}_j}{\sum_{j=1}^{3} \alpha_{1j}} . \tag{37}$$

If the surface vorticity on any triangular patch is assumed to be piecewise linear then the vorticity vector may be written in the general form:

$$\vec{\gamma}_k = (a_0 + a_1 x_k + a_2 y_k)\hat{i} + (b_0 + b_1 x_k + b_2 y_k)\hat{j} , \tag{38}$$

where x_k and y_k lie in the plane of the triangular element. The coefficients (a_0, a_1, a_2) and (b_0, b_1, b_2) are obtained from the following linear equation set:

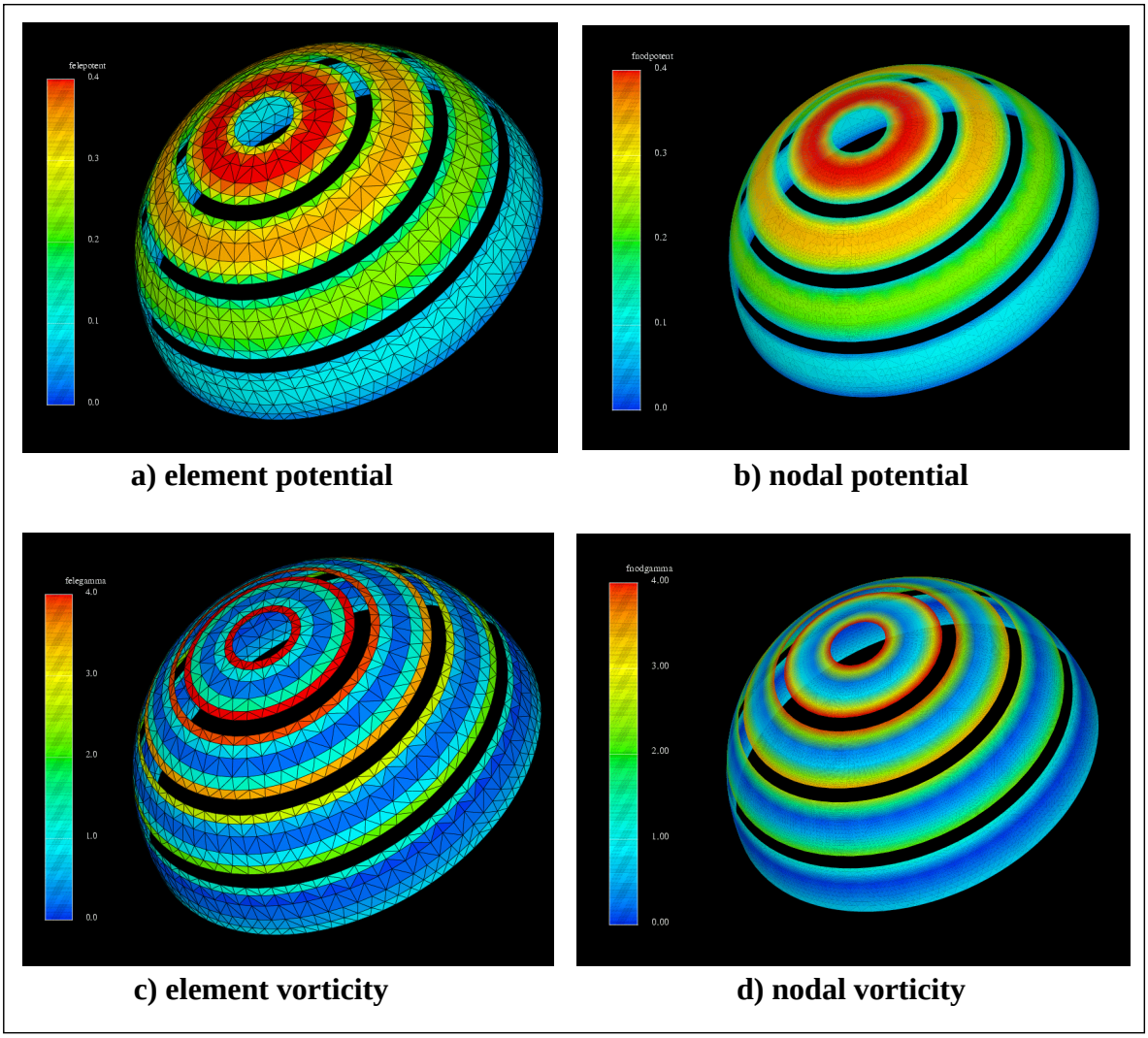
$$\begin{Bmatrix} \vec{\gamma}_{\text{node1}} \\ \vec{\gamma}_{\text{node2}} \\ \vec{\gamma}_{\text{node3}} \end{Bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} \hat{i} + \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \begin{Bmatrix} b_0 \\ b_1 \\ b_2 \end{Bmatrix} \hat{j} , \tag{39}$$

where the $\vec{\gamma}_{\text{node}i}$ values are the averaged vorticity vectors at the three element nodes.

In order for the vorticity vector to be divergence free ($\nabla \cdot \vec{\gamma}_k = 0$) on each element then a_1 plus b_2 must be equal to zero. There is nothing in Equation 39 that guarantees this, although, we should expect it to be approximately true since we have simply smoothed an initially divergence free vorticity field.

It should be noted that while the piecewise linear vorticity distribution is computed in VIPAR_1.0, it is not used in any subsequent computations. On the other hand, the piecewise constant vorticity distribution is used to compute the amount of vorticity that is generated by each surface element.

To summarize the results in this section, we now have a methodology that allows us to smooth the vorticity distribution obtained from a vortex loop boundary element (piecewise constant potential jump) into a piecewise linear vorticity distribution whose magnitude is continuous at the nodes and mating element edges. As an example, the smoothing process is shown for the “MAVEN” geometry in Figure 15.



a) element potential **b) nodal potential**

c) element vorticity **d) nodal vorticity**

Figure 15. Smoothing of the Vortex Loop Boundary Element Method

3.5 Tangential Velocity Boundary Conditions

In order to satisfy the tangential velocity boundary conditions, we use a method similar to that in Reference [13]. The surface vorticity is split into two sheets as indicated in Figure 16 that are placed an infinitesimal distance above and below the membrane surface. The resulting pair of sheets are constructed so as to satisfy both the normal and tangential velocity boundary conditions. These sheets will later move relative to the surface due to convection and diffusion.

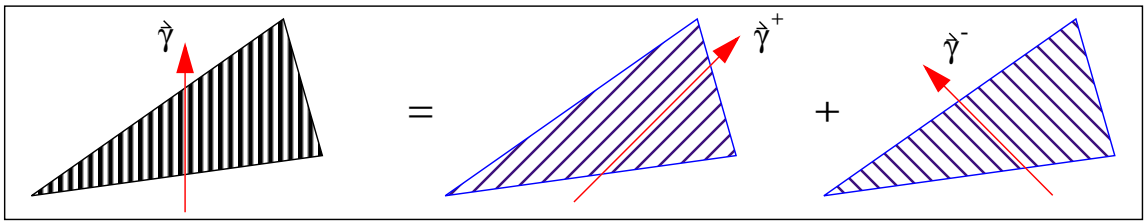


Figure 16. Vorticity Sheet Splitting

In order to compute the vectors $\vec{\gamma}^{+}$ and $\vec{\gamma}^{-}$, expressions are written for the values of the tangential velocities $\vec{U}_{\tau}^{+}$ and $\vec{U}_{\tau}^{-}$ above and below the surface for both the split and original sheets as indicated in Figure 17. Since these velocities must be identical before and after the splitting operation, a formulation may be obtained for the top and bottom sheet strengths as a function of the original sheet strength $\vec{\gamma}$, the local body velocity $\vec{u}_b$, and the local fluid velocity $\vec{u}_f$ induced by the wake and body vortex sheets at the body vortex sheet center.

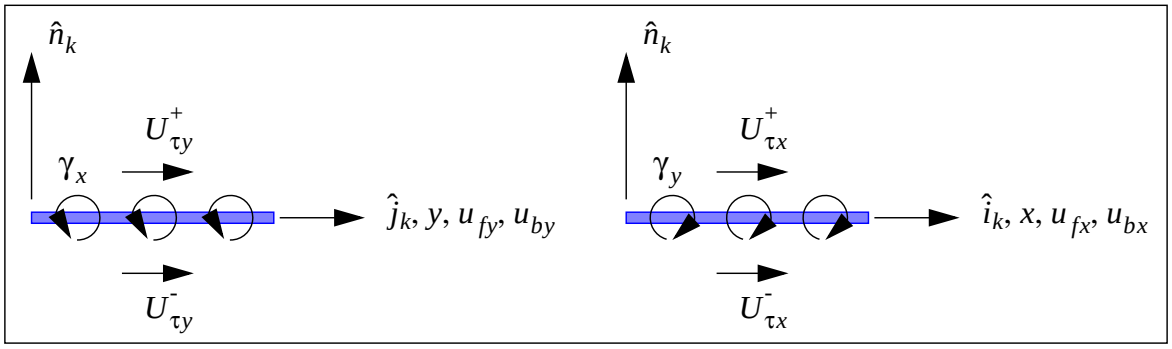


Figure 17. Satisfaction of Tangential Boundary Conditions

We again use the local element coordinate system $(\hat{i}_k, \hat{j}_k, \hat{n}_k)$ where $\hat{i}_k$ and $\hat{j}_k$ are unit vectors that define a rectangular coordinate system in the plane of the surface element and $\hat{n}_k \equiv \hat{i}_k \times \hat{j}_k$. Referring to Figure 17, the velocity vectors on the positive side and negative side of the original vorticity sheet are given by:

$$\vec{U}_{\tau}^{+} = U_{\tau x}^{+}\hat{i}_k + U_{\tau y}^{+}\hat{j}_k, \; \vec{U}_{\tau}^{-} = U_{\tau x}^{-}\hat{i}_k + U_{\tau y}^{-}\hat{j}_k. \qquad (40)$$

The velocity $U_{\tau x}^{+}$ can be obtained by either adding the velocity jump due to the top sheet γ_y^{+} to the tangential body velocity u_{bx} or by adding one-half of the sheet strength γ_y to the local tangential fluid velocity u_{fx} induced by the wake and body vortex sheet. In other words:

$$U_{\tau x}^{+} = u_{bx} + \gamma_y^{+} = u_{fx} + \frac{\gamma_y}{2} \text{ or } \gamma_y^{+} = \frac{\gamma_y}{2} + u_{fx} - u_{bx}. \qquad (41)$$

In a similar fashion, we obtain:

$$\gamma_y^- = \frac{\gamma_y}{2} - (u_{fx} - u_{bx}),\ \gamma_x^+ = \frac{\gamma_x}{2} - (u_{fy} - u_{by}),\ \gamma_x^- = \frac{\gamma_x}{2} + (u_{fy} - u_{by}). \tag{42}$$

The vortex sheet strength vectors for the top and bottom sheet may now be written as:

$$\begin{aligned} \vec{\gamma}^+ &= \left[\frac{\gamma_x}{2} - (u_{fy} - u_{by})\right]\hat{i}_k + \left[\frac{\gamma_y}{2} + (u_{fx} - u_{bx})\right]\hat{j}_k, \\ \vec{\gamma}^- &= \left[\frac{\gamma_x}{2} + (u_{fy} - u_{by})\right]\hat{i}_k + \left[\frac{\gamma_y}{2} - (u_{fx} - u_{bx})\right]\hat{j}_k. \end{aligned} \tag{43}$$

A compact form of Equation 43 for the top and bottom vortex sheets is given by:

$$\vec{\gamma}^{\pm} = \frac{\vec{\gamma}}{2} \pm \hat{n} \times (\vec{u}_f - \vec{u}_b). \tag{44}$$

4 SHEDDING WAKE VORTICITY

4.1 Characterization of Wake Vortex Elements

The vortex elements on the surface (as obtained from the low order formulation) may be characterized as a set of triangular vortex loops that can be smoothed to obtain piecewise linear potential jump distributions on each element (C1 continuity in potential jump). This smoothed distribution will yield piecewise constant surface vorticity on each element (C0 continuity in surface vorticity). The vorticity that is generated during a given time step is shed into the wake during the next time step and the question is: what sort of element is the most suitable to carry this vorticity?

There are several possible choices for wake elements. One could, for example, maintain the integrity (connectivity) of the low order or smoothed sheet and simply diffuse and convect the sheets generated at each time step intact. A second choice would be to keep each triangular element intact. A third choice would be to replace the sheet with a set of disconnected vortons that would mimic the original sheet. A fourth choice is to replace the surface vorticity sheet with a set of vortex rings that are closed on themselves but are disconnected from each other.

The problem with shedding the surface vorticity sheet intact is primarily due to the large amount of book keeping and memory required to maintain connectivity maps for the sheets that are shed from both sides of the membrane surface at each time step. The advantage of such an approach is that no mapping is required to obtain the nascent wake sheet from the surface sheet. In addition, it should be easy to maintain a solenoidal vorticity field on each sheet.

Simply tracking the individual triangular elements eliminates the need to maintain connectivity maps. As a consequence of not having connectivity maps, however, this approach suffers from the fact that many redundant calculations are required. For instance, one must calculate the velocity at the three vertices of each element even though the element shares the vertices with other elements. In addition, the vertices that were originally connected may not remain so due to the accumulation of round-off errors.

Gharakhani [14] recommended that a suitable method for the present work is to model the wake with vorton elements. In this scheme, the surface vorticity sheet that has just been shed would be replaced by a system of vorton or vortex stick elements. Each of these elements convect, stretch, and rotate according to the local velocity field. This requires one to calculate the velocity vector at each end of the element, or compute the velocity vector and its gradient at the vorton center. The most difficult problem with this approach is the maintenance of a solenoidal vorticity field and limitation of the amount of stretch so as to avoid an eventual blowup of the solution.

The fourth choice converts the nascent wake vorticity sheet into a series of vortex rings, loops, or tubes that are closed on themselves but are not connected to each other. Unlike the vortex loops of the low order boundary element formulation, the local tangent to each of these rings represents the direction of the local vorticity vector. The circulation strengths of the vortex loops remain constant as the flow evolves and the vorticity field remains solenoidal. The number of elements (tubes or rings) is greatly reduced although each ring will have a number of nodes in order to resolve its shape. Hairpin removal, filtering of small scale perturbations, and other adaptivity issues should be relatively easy to handle when using this approach. The main disadvantage is the fact that a sometimes complicated mapping between the surface sheet and the nascent wake sheet must be performed.

4.2 Surface Vortex Element Conversion to Wake Vortons

In VIPAR_1.0 we have chosen to convert the surface vorticity on each triangular element into an associated set of vortons. Each triangular element consisting of a constant vortex patch may be approximated by a single vorton element whose strength $\vec{\Upsilon}$ is equal to the sheet vorticity $\vec{\gamma}$ times the element area δA or equivalently to a circulation Γ times a length $\vec{\delta l}$ as indicated in Figure 18.

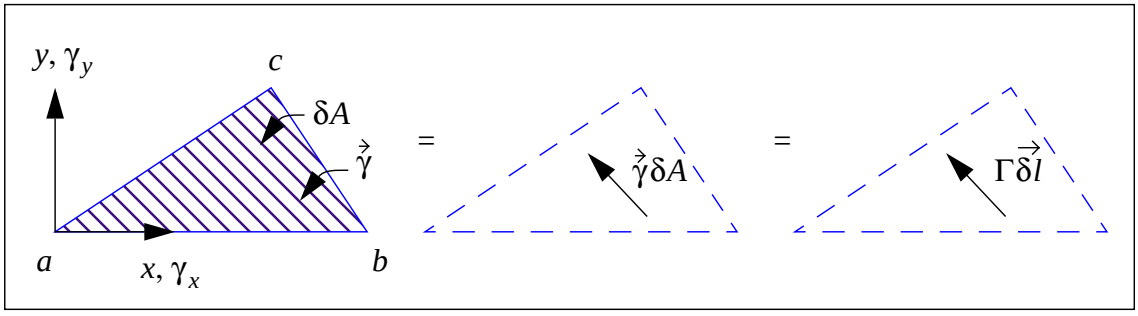


Figure 18. Surface Vortex Patch to Wake Vorton

The velocity field $\vec{u}(\vec{r})$ at any point $\vec{r}$ in the field induced by this element is given by:

$$\vec{u}(\vec{r}) = \nabla \times \int_A \frac{\vec{\gamma}}{4\pi|\vec{r}-\vec{r}'|} dA \approx \nabla \times \left(\frac{\vec{\Upsilon}\xi_\Phi}{4\pi|\vec{r}-\vec{r}'|} \right) = \nabla \times \left(\frac{\Gamma \vec{\delta l}\xi_\Phi}{4\pi|\vec{r}-\vec{r}'|} \right). \qquad (45)$$

Here, ξ_Φ is a core function and $\vec{r}'$ is the position of the center of the vorton.

As will be shown in the next section, the evolution equation for $\vec{\Upsilon}$ satisfies the “stretching term” in the vorticity transport equation and is given by:

$$\frac{D\vec{\Upsilon}}{Dt} = (\vec{\Upsilon} \cdot \nabla)\vec{u}(\vec{r}). \qquad (46)$$

If the vorton strength is represented by $\Gamma\vec{\delta l}$, then the circulation may be held constant and the length $\vec{\delta l}$ allowed to vary according to the evolution equation

$$\frac{D\vec{\delta l}}{Dt} = (\vec{\delta l}\cdot\nabla)\vec{u}(\vec{r}). \tag{47}$$

Equation 47 may be satisfied by simply convecting the ends of the “stick” at the local fluid velocity.

The initial choice for δl is somewhat arbitrary but should be on the order of the initial element size. One such choice is the following:

$$\frac{\pi(\delta l)^2}{4} = \delta A = \frac{1}{2}(x_b - x_a)(y_c - y_a), \tag{48}$$

$$\delta l = \sqrt{\frac{2}{\pi}(x_b - x_a)(y_c - y_a)},$$

where the subscripts a, b, c refer to the element vertices shown in Figure 18. The circulation strength Γ is then given by:

$$\Gamma = \frac{\delta A}{\delta l}|\gamma| = \frac{\pi}{4}\delta l|\gamma| = |\gamma|\sqrt{\frac{\pi}{8}(x_b - x_a)(y_c - y_a)}. \tag{49}$$

In order for the velocity perturbations from the nascent vortons on the surface to be smooth, the vorton centers should be located approximately $\delta l/2$ away from the wall. In order for the effects of the Gaussian core function to be small at the surface, the core radius should be on the order of $\delta l/3$.

(this page intentionally left blank)

5 EVOLUTION OF WAKE VORTICITY

5.1 Governing Equations

The vorticity equation resulting from taking the curl of the constant property Navier-Stokes equation is given by:

$$\frac{D\vec{\omega}}{Dt} = (\vec{\omega}\cdot\nabla)\vec{u} + \nu\nabla^2\vec{\omega}. \tag{50}$$

From Equation 50 we see that the vorticity vector $\vec{\omega}$ can be stretched and diffused.

5.2 Vortex Stretching

Expanding upon the discussion in Section 4.2 on page 34, we consider the contribution to the time rate of change of vorticity from the stretching term in Equation 50:

$$\left.\frac{D\vec{\omega}}{Dt}\right|_{\text{stretch}} = (\vec{\omega}\cdot\nabla)\vec{u}. \tag{51}$$

Integrating both sides of Equation 51 over a volume consisting of a vortex tube segment with a vector length of $\vec{\delta l}$ radius σ, and circulation Γ yields:

$$\int_{V_i}\left(\left.\frac{D\vec{\omega}}{Dt}\right|_{\text{stretch}}\right)dV = \int_{V_i}((\vec{\omega}\cdot\nabla)\vec{u})dV, \tag{52}$$

$$\Gamma\frac{D\vec{\delta l}}{Dt} = \int_{V}\left(\left.\frac{D\vec{\omega}}{Dt}\right|_{\text{stretch}}\right)dV, \tag{53}$$

$$\Gamma(\vec{\delta l}\cdot\nabla)\vec{u} = \int_{V}((\vec{\omega}\cdot\nabla)\vec{u})dV. \tag{54}$$

Combining Equations 52, 53, and 54 we obtain Equation 47 repeated here as:

$$\frac{D\vec{\delta l}}{Dt} = (\vec{\delta l}\cdot\nabla)\vec{u}(\vec{r}). \tag{55}$$

From Batchelor [15], we see that this equation is satisfied in the limit as $\vec{\delta l}\to 0$ so long as the ends of the vortex tube of vector length $\vec{\delta l}$ convect at the local fluid velocity. Furthermore, the time rate of change of vorticity from stretching can be expressed by:

$$\left.\frac{D\vec{\omega}}{Dt}\right|_{\text{stretch}} = \frac{\Gamma}{\pi\sigma^2\delta l}(\vec{\delta l}\cdot\nabla)\vec{u}(\vec{r}) = \frac{\Gamma}{\pi\sigma^2\delta l}\frac{D\vec{\delta l}}{Dt}, \tag{56}$$

where δl is the magnitude of $\vec{\delta l}$.

Since the stretching term of Equation 51 is satisfied by simply convecting the ends of the vortex tube element of length δl and radius σ then the time rate of change of the core radius σ can be related to that of the element length through continuity ($\delta l \sigma^2 = \text{constant}$):

$$\left.\frac{d\sigma}{dt}\right|_{\text{stretch}} = -\frac{1}{2}\frac{\sigma}{\delta l}\frac{d\delta l}{dt}. \tag{57}$$

5.3 Diffusion of Vorticity

Next we consider the contribution to the time rate of change of vorticity from the diffusion term in Equation 50:

$$\left.\frac{D\vec{\omega}}{Dt}\right|_{\text{diffusion}} = \nu\nabla^2\vec{\omega}. \tag{58}$$

The solution of Equation 58 for a vortex tube is given by:

$$\vec{\omega} = \frac{\vec{\Gamma}}{4\pi\nu t}e^{-\frac{r^2}{4\nu t}}, \tag{59}$$

for the initial condition $\vec{\omega}(0) = \infty$. Equation 59 can also be written as:

$$\vec{\omega} = \frac{\vec{\Gamma}}{\pi\sigma^2}e^{-\frac{r^2}{\sigma^2}}, \tag{60}$$

with the implication that the core radius is given by $\sigma = \sqrt{4\nu t}$. Of more interest is the implied rate of growth of the core radius:

$$\left.\frac{d\sigma}{dt}\right|_{\text{diffusion}} = \frac{1}{2}\sqrt{\frac{4\nu}{t}} = \frac{2\nu}{\sigma}. \tag{61}$$

This is the so called “diffusing or expanding core method” that has been used to account for diffusion in 2D vortex methods. Since the diffusion equation is linear, vortex elements that have a vorticity distribution given by Equation 60 may be diffused independently and then superimposed. While the diffusion solution is exact, the vorticity in the flow may not be convected properly especially if the core radii are allowed to become large.

5.4 Core Growth

The time rate of change of the radius σ of a vortex tube segment with length δl as affected by diffusion and stretch is obtained by combining Equations 57 and 61:

$$\frac{d\sigma}{dt} = \frac{2\nu}{\sigma} - \frac{1}{2}\frac{\sigma}{\delta l}\frac{d\delta l}{dt}. \tag{62}$$

Here, the first term contributes the diffusive effect while the second term is due to stretch. The time rate of change of volume for the tube segment is:

$$\frac{dV}{dt} = \pi\frac{d(\sigma^2\delta l)}{dt} = \pi\sigma\left(2\delta l\frac{d\sigma}{dt} + \sigma\frac{d\delta l}{dt}\right). \tag{63}$$

Substituting Equation 62 into Equation 63 yields:

$$\frac{dV}{dt} = 4\pi\delta l\nu\,. \tag{64}$$

If the vortex tube is represented by a single vorton with core radius $\tilde{\sigma}$ then the time rate of change of volume for the vorton is:

$$\frac{dV}{dt} = \frac{4}{3}\pi\frac{d\tilde{\sigma}^3}{dt} = 4\pi\tilde{\sigma}^2\frac{d\tilde{\sigma}}{dt}\,. \tag{65}$$

Combining Equations 64 and 65 yields:

$$\frac{d\tilde{\sigma}}{dt} = \left(\frac{\delta l}{\tilde{\sigma}}\right)\frac{\nu}{\tilde{\sigma}}\,. \tag{66}$$

If one approximates the quantity $\delta l/\tilde{\sigma}$ as a constant equal to $\delta l_o/\tilde{\sigma}_o$ during a time step then integration over that time step $(t - t_o)$ gives the following results:

$$\tilde{\sigma} \approx \sqrt{\tilde{\sigma}_o^2 + \left(\frac{\delta l_o}{\tilde{\sigma}_o}\right)4\nu(t - t_o)}\,. \tag{67}$$

Thus we see that an increase in $\delta l_o/\tilde{\sigma}_o$ due to stretching enhances the diffusion process.

Since we do not compute the vorticity field explicitly in VIPAR_1.0, the primary function of the vorton core radius $\tilde{\sigma}$ is to regularize the vector potential and velocity field calculations. The vorticity field could be calculated but we know that the field will not be smooth due to the lack of core overlap of the nascent vortons that are formed at the surface. Not overlapping the nascent vorton cores is a compromise associated with surface grid resolution, nascent vorton placement (standoff distance) and the influence of core functions across impermeable boundaries. Eventually, we hope to be able to alleviate this problem as we are better able to model the boundary with increased resolution, higher order boundary elements, larger numbers of nascent vortons, and perhaps specialized nascent vortex elements.

(this page intentionally left blank)

6 CONVECTION OF WAKE ELEMENTS

6.1 Vector Potential

The vector potential at $\vec{r}$ due to a vorton at $\vec{r}'$ is given by:

$$\vec{\Phi} = \frac{\Gamma \delta l}{4\pi |\vec{r} - \vec{r}'|} \xi_\Phi \hat{e}_{\delta l}, \tag{68}$$

where $\hat{e}_{\delta l}$ is the unit vector in the direction of $\vec{\delta l}$ and ξ_Φ is a core function that regularizes the expression for $\vec{\Phi}$ when $r = 0$ (i.e. Φ should be finite). As given by Strickland and Baty [16], the core function is:

$$\xi_\Phi = 1 - e^{-q\rho}, \tag{69}$$

$$q = 1.354 + 0.842\rho + 0.559\rho^2, \tag{70}$$

$$\rho \equiv \frac{|\vec{r} - \vec{r}'|}{\tilde{\sigma}}. \tag{71}$$

It should be noted that the core function ξ_Φ approaches unity as one moves away from the vorton center by several core radii.

6.2 Velocity Calculations

The velocity at $\vec{r}$ due to a vorton at $\vec{r}'$ can be obtained from the vector potential as:

$$\vec{u}(\vec{r}) = \nabla \times \vec{\Phi}. \tag{72}$$

Strickland and Baty [16] used a working expression for $\vec{u}(\vec{r})$ based on the components of the vorton strength $\vec{\Upsilon} \equiv \Gamma \vec{\delta l}$ and the distance $\vec{r} - \vec{r}'$ given by:

$$u_i(\vec{r}) = \frac{\xi_u}{4\pi |\vec{r} - \vec{r}'|^3} [(\vec{r} - \vec{r}')_k (\Gamma \delta l)_j - (\vec{r} - \vec{r}')_j (\Gamma \vec{\delta l})_k], \tag{73}$$

$$\xi_u = 1 - e^{-\rho^3}. \tag{74}$$

It should be noted that the indices (i, j, k) represent the components in the (x, y, z), (y, z, x), or (z, x, y) coordinate directions. The velocity core function $\xi_u(\rho)$ has the property that will cause the velocity to be equal to zero for $\rho = 0$. The core function itself will approach unity for $|\vec{r} - \vec{r}'|$ greater than several σ.

In order to perform fast yet accurate calculation of the velocity field from the vorticity field, we have used a fast multipole method due to Strickland, Gritzo, Baty, Homicz, and Burns [17]. Overviews of various fast multipole schemes and their applicability to a large number of physics problems are given by Greengard [18] and by Strickland and Baty [19]. For many problems, the proper discretization of the source field (vorticity) and target field (velocity) will require the use of more than $N_S = 10^6$ source particles and $N_T = 10^6$ target points. In order to perform efficient calculations in such cases, the combined use of fast multipole expansions and massively parallel computing is required. We therefore sought a solution methodology that provided good "parallel efficiency," while at the same time reducing the CPU time from order $O(N_S N_T)$ to something approaching $O(\sqrt{N_S N_T})$. Results from the scheme reported in [17] and [20] indicate that good parallel efficiencies are obtained when there are at least 1,000 vortons per processor. The multipole method breaks even with the direct calculation when there are on the order of 70,000 vortons in the flow.

We note that the recently developed three-dimensional fast multipole method of Cheng, Greengard, and Rokhlin [21] that uses local expansions has a break even point on the order of 1,000 vortons and is approximately two orders of magnitude faster than our method for 10^6 vortons. Our aversion to using the "Greengard-Rokhlin Scheme" is that parallel efficiencies scale poorly for this scheme. Thus for parallel computing where several thousand processors are available, our present choice is appropriate.

6.3 Vorton Displacements

The vortons that represent the vorticity in the wake induce velocities on each other according to Equation 73 or from the fast multipole solver. The vector velocity $\vec{u}(\vec{r})$ at any point $\vec{r}$ can be expressed by:

$$\vec{u}(\vec{r}) = \sum_{i=1}^{N_v} \vec{u}_i(\vec{r}) \tag{75}$$

where, $\vec{u}_i(\vec{r})$ is the induced velocity vector at point $\vec{r}$ due to a vorton at point i and N_v is the total number of vortons in the flow field. Each vorton in the flow is convected at the local fluid velocity $\vec{u}(\vec{r})$. The distance that a particular vortex moves between time $t_0 = t - \Delta t$ and time $t_1 = t$ is obtained using the following second order Adams-Bashforth open integration formula [22]:

$$\Delta\vec{r}_V = \left[\frac{3}{2}\vec{u}(\vec{r}_V)\Big|_{t_1} - \frac{1}{2}\vec{u}(\vec{r}_V)\Big|_{t_0}\right]\Delta t\,. \tag{76}$$

Here, $\Delta\vec{r}_V$ is the change in the position vector associated with the center of the vorton while $\vec{u}(\vec{r}_V)\big|_{t_1}$ and $\vec{u}(\vec{r}_V)\big|_{t_0}$ are the vector velocities of the vorton center at the current and previous time steps respectively. A simple Euler integration is used for the first time step after the nascent vorton is created.

In VIPAR_1.0 we actually convect the head and tail of the vorton element as indicated in Figure 19 and average the results to determine the new vorton position. This also allows the vorton vector length to stretch and rotate in the flow field.

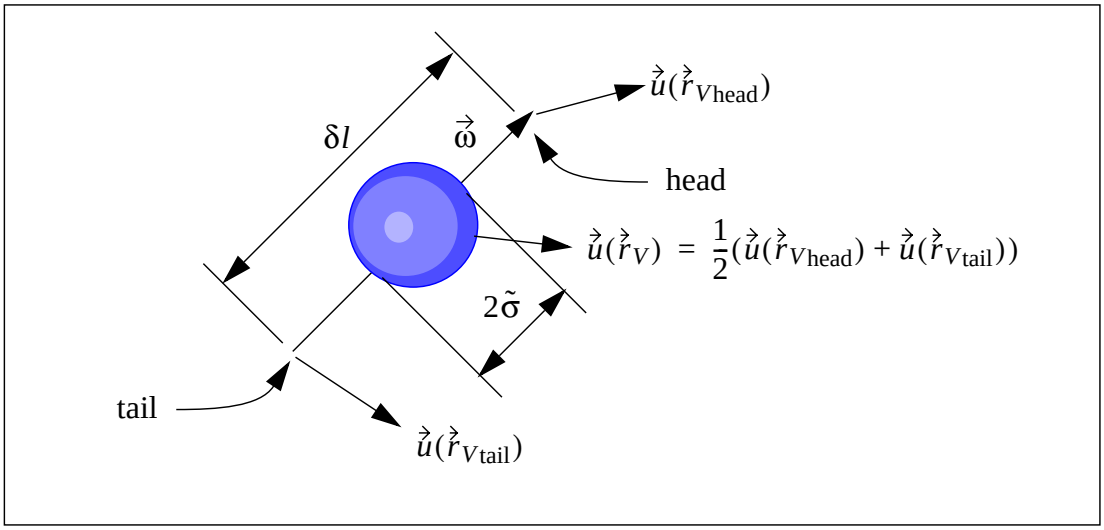


Figure 19. Motion of Vorton Element

An alternative approach for obtaining the amount of stretch and rotation is to compute the gradient of the velocity field at the vorton center $\nabla\vec{u}(\vec{r}_V)$ using perhaps a moving least squares (MLS) method. This of course requires identification of near neighbors to each vorton. After obtaining the velocity gradient $\nabla\vec{u}(\vec{r}_V)$, the head and tail velocities can then be computed from:

$$\vec{u}(\vec{r}_{V\text{head}}) = \vec{u}(\vec{r}_V) + (\vec{\delta l}\cdot\nabla)\vec{u}(\vec{r}_V), \tag{77}$$

$$\vec{u}(\vec{r}_{V\text{tail}}) = \vec{u}(\vec{r}_V) - (\vec{\delta l}\cdot\nabla)\vec{u}(\vec{r}_V).$$

New positions of the head and tail can now be obtained from Equation 76. The major advantage of this scheme is that the stretching and rotation are based on head and tail velocities that are obtained using the MLS method that involves a number of near neighbors. This should provide a much smoother and possibly more accurate set of results. Also, only one velocity calculation $\vec{u}(\vec{r}_V)$ is required for each vorton. This later advantage is offset by the need to compute the gradient $\nabla\vec{u}(\vec{r}_V)$.

(this page intentionally left blank)

7 COMPUTING ENVIRONMENT

7.1 I/O Data Base

Although the fluid mechanic equations are solved using a gridless vortex method, the dynamics of the canopy are simulated using a finite-element (FE) model made up of triangular elements. Therefore, a simulation database is used that can handle FEs, since the wake vortices themselves may be thought of as single-noded FEs.

FE databases can be very large; storing them in binary, as opposed to text format, can save quite a lot of disk storage. However, binary files are generally not compatible between different computers. In an effort to bridge these seemingly contradictory requirements, Sandia developed the EXODUS II Library [23], hereafter referred to simply as EXODUS. EXODUS defines a standardized random-access model for storing finite-element data that is code- and machine-independent, and provides a set of Application Program Interface (API) routines that can be used to write and/or read EXODUS databases. The API routines are callable from Fortran, C, and C++. These routines make use of the low-level netCDF Software [24] to assure that the resulting database is both binary and machine-independent, *i.e.*, the same file can be read by computers that use different means to store integers, characters, and floating-point numbers.

The commercial package PATRAN™ is used for the initial surface meshing and creates a FE model of the canopy that can be written out as an EXODUS database from within PATRAN™. VIPAR itself uses the EXODUS database format to write out all simulation results. The results can be visualized using another commercial package, EnSight™ [2], that is capable of reading EXODUS files. In the future we expect to implement a “restart” capability in VIPAR, as insurance against system crashes. The use of EXODUS restart files will not only save storage, but allow the continuation of a simulation on a platform other than the one on which it was begun.

7.2 Parallel Computing

VIPAR_1.0 was parallelized since it is anticipated that VIPAR simulations of full-scale systems will involve on the order of 10^4–10^6 surface elements on the canopy and 10^6–10^8 vortices in the surrounding air. This requires the use of massively parallel architectures and algorithms to keep the run-time reasonable.

The best single indication of performance is the parallel efficiency, η, defined as:

$$\eta = \frac{t(1)}{P\,t(P)} \tag{78}$$

where $t(P)$ is the wall-clock time taken by P processors. Thus, an algorithm that took 100 hr. to solve a problem on one processor, and 1 hr. to solve it using 100 processors, would have a parallel efficiency of 1.0.

Homicz and Burns [20] parallelized the first VIPAR code module that was the fast multipole solver that computes the velocity field from the vorticity field. This code module has a major

impact on VIPAR CPU time. As shown in Figure 20, very good parallel efficiencies have been demonstrated on the ASCI Red TeraFlops computer [25]. The efficiency is better than 80% with at least 1,000 vortex elements held by each processor, and in excess of 90% with 10,000 or more vortices/processor. The algorithm also exhibits excellent scalability, having been run for model problems with up to 8,000,000 vortex elements, using as many as 2,048 processors.

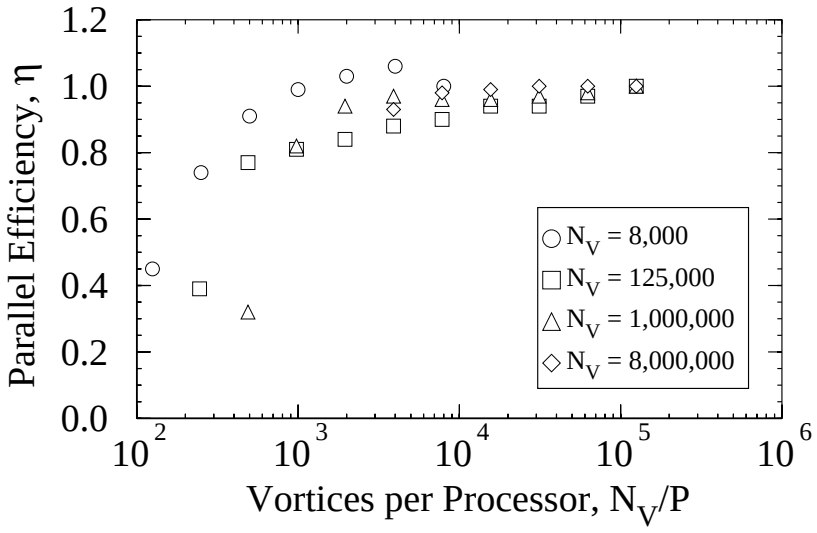


Figure 20. Parallel Efficiency *vs.* Vortex Population on Each Processor (fast multipole solver)

Much of the computational work in VIPAR involves looping over surface elements, nodes, or the vortices in the surrounding fluid. Thus a parallelization strategy that distributes these entities evenly across processors tends to result in a more equitable partition of work.

In order to split the surface elements across processors, we first generate a “global” finite-element (FE) model of the canopy surface using PATRAN™, and write it out to an EXODUS database. A library of routines developed at Sandia, NEMESIS [26], takes this file as input along with the number of processors, *P*, to be used in the simulation, and splits it into *P* separate “local” subdomain models. Each consists of a set of contiguous surface elements and nodes, that, to the extent possible, NEMESIS apportions equally between the subdomains. Each subdomain is written to a separate EXODUS database and assigned as the input geometry to a corresponding processor. The nodes and elements remain stored on that processor for the duration of the simulation, thus assuring that their storage, and any computational work that scales with the number of nodes and elements, is evenly distributed across the processors.

Further, NEMESIS performs the splitting in such a way that the number of nodes shared by neighboring subdomains is minimized. The amount of inter-processor communication is usually proportional to the number of nodes shared between processors, so in effect this minimizes the communication overhead, thus increasing efficiency. Inter-processor communication in VIPAR is performed using the Message Passing Interface (MPI)

Standard [27]. MPI Libraries written to this standard are now available on most, if not all, parallel platforms, leading to the quick acceptance of MPI as the *de facto* message-passing library of choice in codes where portability is an important issue.

Finally, as was indicated earlier, two vortex elements are shed from each surface element at each time step. These vortices remain stored on the same processor that stores the corresponding surface element. Since the elements are evenly distributed, so will be the memory requirements associated with the vortex population, as well as any computational work that scales with the number of vortices.

To avoid the conflicts engendered by having multiple processors attempting to write to the same file, each processor outputs results for its local population of elements, nodes, and vortices to its own unique set of two files. Surface positions, velocities, *etc*., are written to a local "surface" output file, while vortex positions, strengths, *etc*., are written to a local "wake" file. Results from these local files are rejoined using NEMESIS into two global files (a wake file and a surface file) that may be viewed by EnSight™ [2] either separately or together.

7.3 Portability

A paramount goal of the ASCI program is the development of computational tools that are portable across the wide spectrum of computers available in the Nuclear Weapons Complex (NWC). VIPAR is written to conform with the Fortran 77 Standard. As the bulk of the source code is common to serial and parallel operation, a single source code is maintained for both. C-preprocessor directives, recognized by most Fortran compiler packages, are used to include/exclude blocks of code at compile time, according to whether a serial or parallel executable image is specified in the command line. These directives are also used to specialize those sections of code that are likely to change from one computer platform to the next, *e.g.*, I/O file systems.

(this page intentionally left blank)

8 FLUID STRUCTURE COUPLING

8.1 General Description

The coupling of the fluids code VIPAR with the structural dynamics code PRESTO has recently been accomplished. We provide here only a brief summary of the coupling algorithm in that a subsequent report will provide a more complete description. The coupling occurs in the context of the SIERRA software development environment (Taylor [28] et al.). This environment consists of an object-oriented framework of software services that are common to Sandia's suite of mechanics application codes and a set of software tools for managing the development process.

The structural dynamics code PRESTO is the SIERRA version of PRONTO3D that was originally developed by Taylor and Flanagan [29]. Modifications were made in PRESTO in order to account for the behavior of textile material and to interface smoothly with the VIPAR code.

8.2 Added Mass Coupling Algorithm

Since the parachute mass is relatively small as compared to the mass of fluid that is accelerated during deployment, VIPAR and PRESTO must be tightly coupled. It is not sufficient for VIPAR to simply pass surface pressures to PRESTO and for PRESTO to pass surface shape and velocity back to VIPAR. Some representation of the force required to accelerate the "added mass" of fluid adjacent to the body surface must be included in the structural calculation in order to keep the calculation both accurate and stable. This is made more necessary by the fact that PRESTO is an explicit code and many structural time steps are required for each fluid time step.

Ideally, one would use a full "fluid added mass matrix" in the structural code. However, we have found this to be unnecessary as well as impractical. Thus, the stabilization scheme that we have chosen is to add fluid mass to each structural node, this mass being acted upon by body acceleration normal to the surface. These fluid masses are obtained by diagonalizing the full added mass matrix. In the present scheme, the full matrix is diagonalized by conserving the total normal surface force. This is equivalent to summing the columns of the added mass matrix and lumping those sums on the diagonal.

For the purposes of transferring the added mass information from VIPAR to PRESTO, it is advantageous to compute the added mass per unit surface area $\{\tilde{m}_j\}$. The vector $\{\tilde{m}_j\}$ is simply the diagonal of the diagonalized added mass matrix per unit area. In order to obtain $\{\tilde{m}_j\}$, the following linear equation is solved at the beginning of each fluid time step:

$$\frac{1}{\rho}\frac{\Delta t_f}{\Delta t_s}[A_i]^{-1}[A_{ij}]^T[A_i]\{\tilde{m}_j\} = \{1\}. \tag{79}$$

Here, Δt_f and Δt_s are the sizes of the fluid and structural time steps respectively. The diagonal matrix $[A_i]$ represents the element surface areas, $[A_{ij}]$ is the coefficient matrix of Equation 14, and ρ is the fluid density.

8.3 Fluid Pressure at Structural Time Steps

The fluid pressure at the beginning of the n^{th} structural time step is:

$$\Delta p_i(t_f + \Delta\tau_s(n)) \;=\; \Delta p_i(t_f) - \sum_{k=1}^{n} \tilde{m}_i \, \dot{u}_{bni}(t_f + \Delta\tau_s(k-1)) \qquad (80)$$

where $\Delta\tau_s(n) \;=\; \sum_{k=1}^{n} \Delta t_s$, and $\dot{u}_{bni}$ is the normal acceleration of the i^{th} surface element.

Heuristic evidence from a 1D fluid structure model indicates that this coupling scheme will provide stability as well as an accurate solution even if the added mass is overestimated by an order of magnitude. The solution accuracy for the multi-degree-of-freedom case, using the diagonalized fluid mass matrix, will be formally evaluated.

9 PRELIMINARY RESULTS

9.1 Rigid Geometries

In this section, we present a set of preliminary results from the parallelized VIPAR code that appear in an earlier paper by Strickland, Homicz, Gossler, and Porter [30]. The body geometries chosen are a 3-ring ringslot parachute, a cross parachute, and a partial sphere with vent. These geometries were constructed on a spherical surface. The bodies, that are assumed to be rigid, were started impulsively and translated along their axis of symmetry at a constant velocity U_∞. In these preliminary runs, approximately 700 surface elements are used and the code is run on a small number of processors (8). Each run begins impulsively and runs for about 80 time steps. During a single time step, the body moves 1/10 of a body radius, where a body radius is the radius of the sphere on which the geometries were formed.

9.1.1 3 Ring Ringslot

Results are obtained for a 3 ring ringslot parachute shape formed on a sphere as indicated in Figure 21. The ring closest to the vent extends from a latitude of 81 degrees North to a latitude of 58 degrees North. The second ring extends from a latitude of 52 degrees North to a Latitude of 29 degrees North. The skirt ring lies between 23 degrees North and the equator. The surface was discretized into 695 triangular elements with 415 nodes.

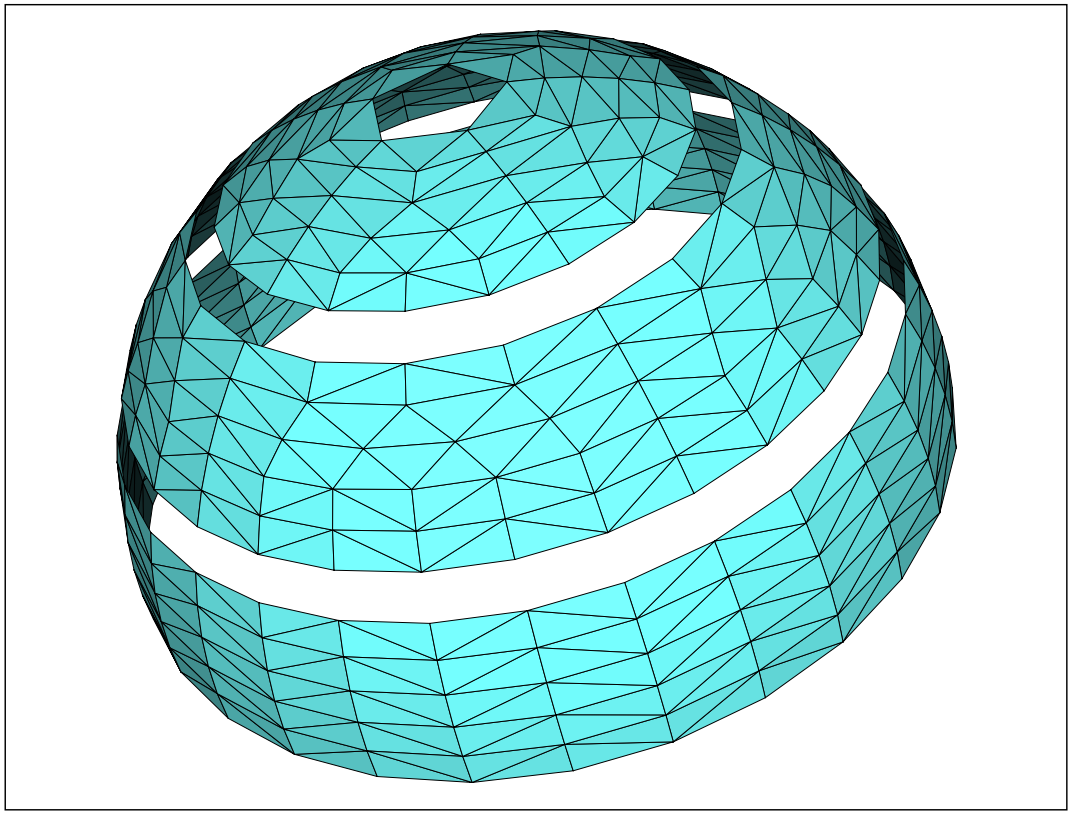
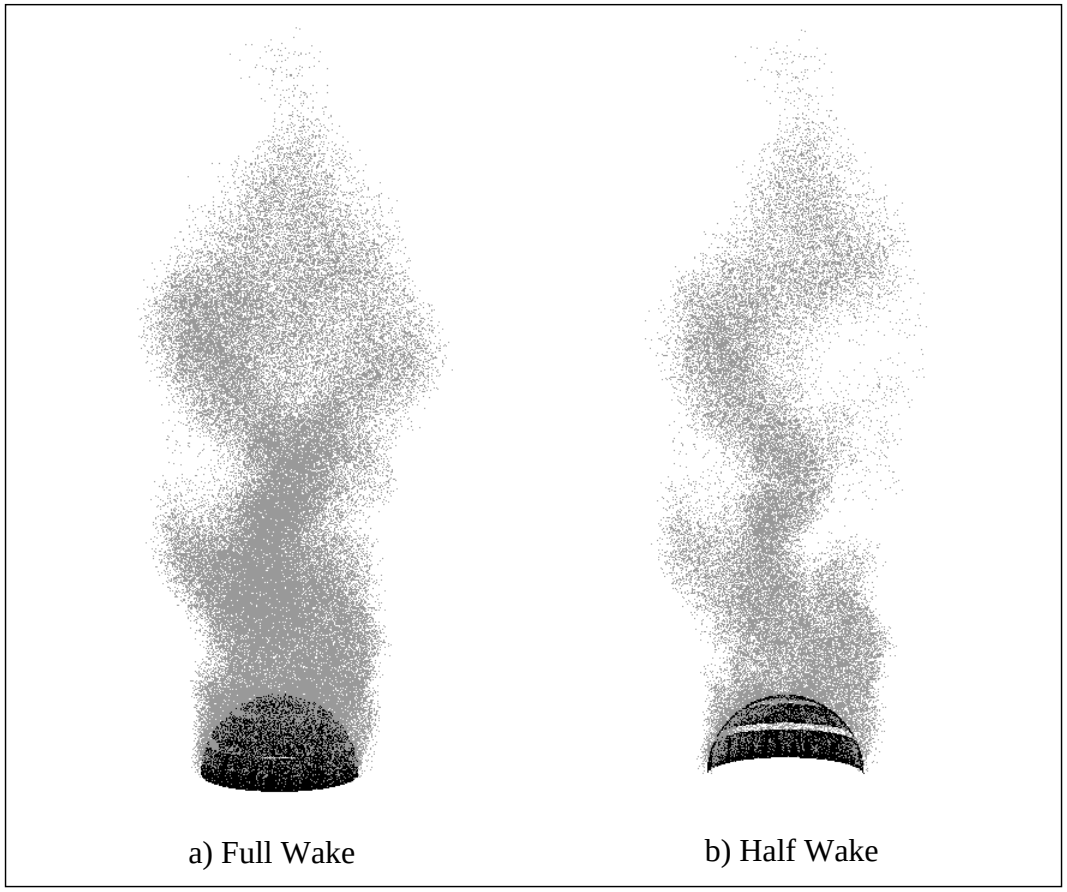


Figure 21. Surface Mesh for 3 Ring Ringslot

A snapshot of the wake behind the 3 ring ringslot at a non dimensional time tU_∞/R of 7.0 is shown in Figure 22. Here, R is the radius of the sphere. The "half wake" shown in Figure 22b is simply the back half of the full wake shown in Figure 22a. It is clear from this figure that the wake has become highly three-dimensional.



a) Full Wake b) Half Wake

Figure 22. Wake Behind 3 Ring Ringslot

In Figure 23, the axial force coefficient time history is shown along with data given in the Recovery Systems Design Guide [31] for ringslots constructed from a flat circular plan form. The conversion between the drag coefficient C_{Do} given in the design guide (0.56-0.65 for a flat ringslot) and C_y in Figure 23 is given by:

$$\frac{C_y}{C_{Do}} = \frac{S_o}{R^2} = 6.53, \tag{81}$$

where S_o is the total decelerator surface area on a flat plan form and R is the sphere radius.

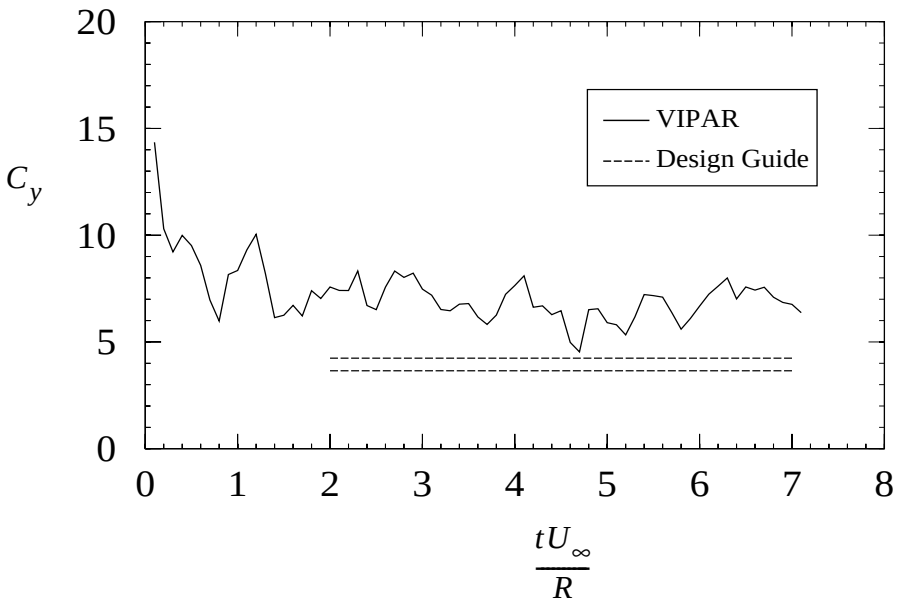


Figure 23. Axial Force Coefficient for 3 Ring Ringslot

The drag coefficient predicted using the VIPAR spherical parachute shape only roughly approximates the drag coefficient for the design guide parachute shape constructed from a flat circular panel. The actual inflated shape would be somewhat different and would of course include gores. VIPAR over-predicts the drag coefficient by about 50%. There are probably several reasons for this, including minimal spatial and temporal resolution. For example, the drag force on an impulsively started flat plate will be over-predicted by about 30% using the low order boundary element scheme (used in all of these simulations) if the plate only contains four or five elements across its width. The lack of sufficient temporal resolution, on the other hand, can cause the wake to become less coherent that in turn, requires stronger vorticity to be generated on the surface to satisfy the normal surface boundary condition and counterbalance the oncoming freestream. Thus, it is essential that the flow be properly resolved that will require the full use of our largest massively parallel computers.

9.1.2 Rotating Cross Parachute

Results are obtained for a cross parachute formed on a sphere, shown in Figure 24. The width of the strips making up the cross are equal to 25% of the sphere diameter. The strips have aspect ratios of 7:1. The ends of the strips therefore are at a latitude of 10.27 degrees South. The surface was discretized into 650 triangular elements with 396 nodes.

In addition to the impulsively started translational motion, the cross parachute in this particular simulation rotates about the polar axis at a non dimensional rate of $\Omega R/U_\infty = 0.1$.

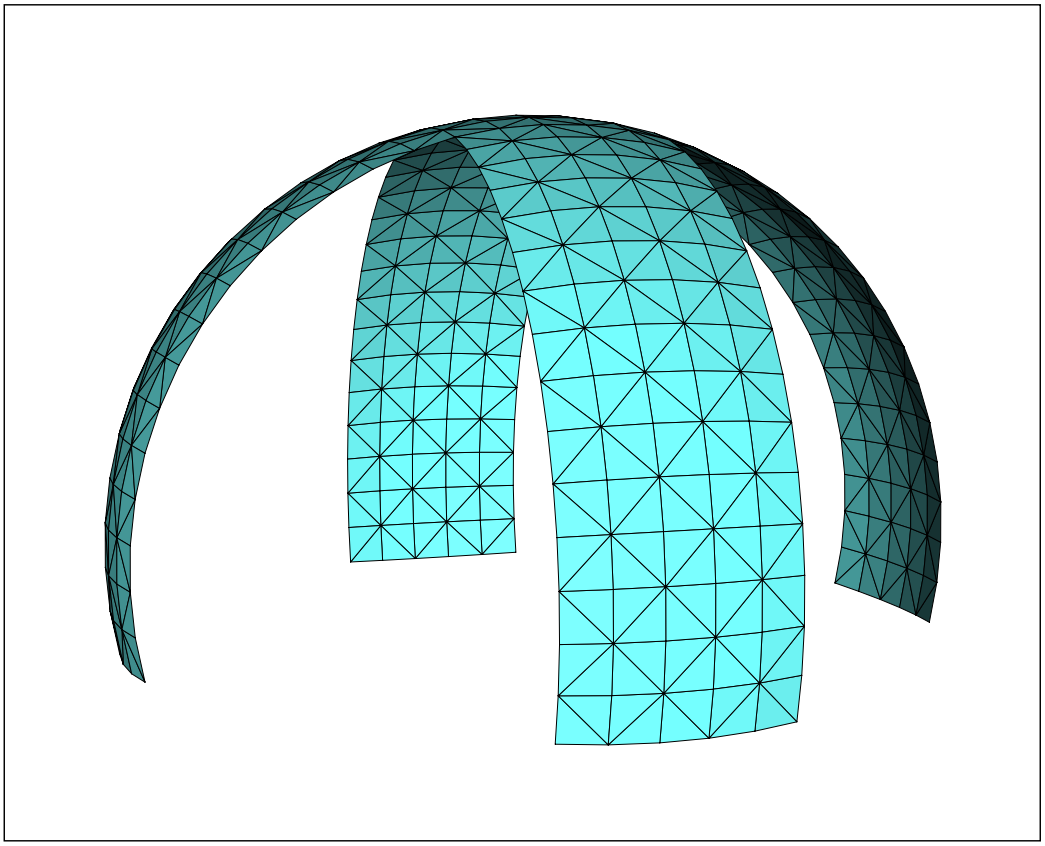


Figure 24. Surface Mesh for Cross Parachute

A snapshot of the wake behind the cross parachute at a non dimensional time tU_∞/R of 8.0 is shown in Figure 25. The corkscrew pattern of the wake induced by rotation can be inferred by comparing Figures 25a and 25b.

In Figure 26, the axial force coefficient time history is shown along with data given in the Recovery Systems Design Guide [31] for cross parachutes. The conversion between the drag coefficient C_{Do} given in the design guide and C_y in Figure 25 is again given by Equation 81 with $S_o/R^2 = 3.25$. For this case, C_{Do} design guide values range from 0.66 to 0.72. The VIPAR prediction is closer to that given in the design guide for a cross parachute than it was for the ringslot although it is still high by at least 30%. Accuracy in the drag coefficient prediction is expected to improve with increased spatial and temporal resolution.

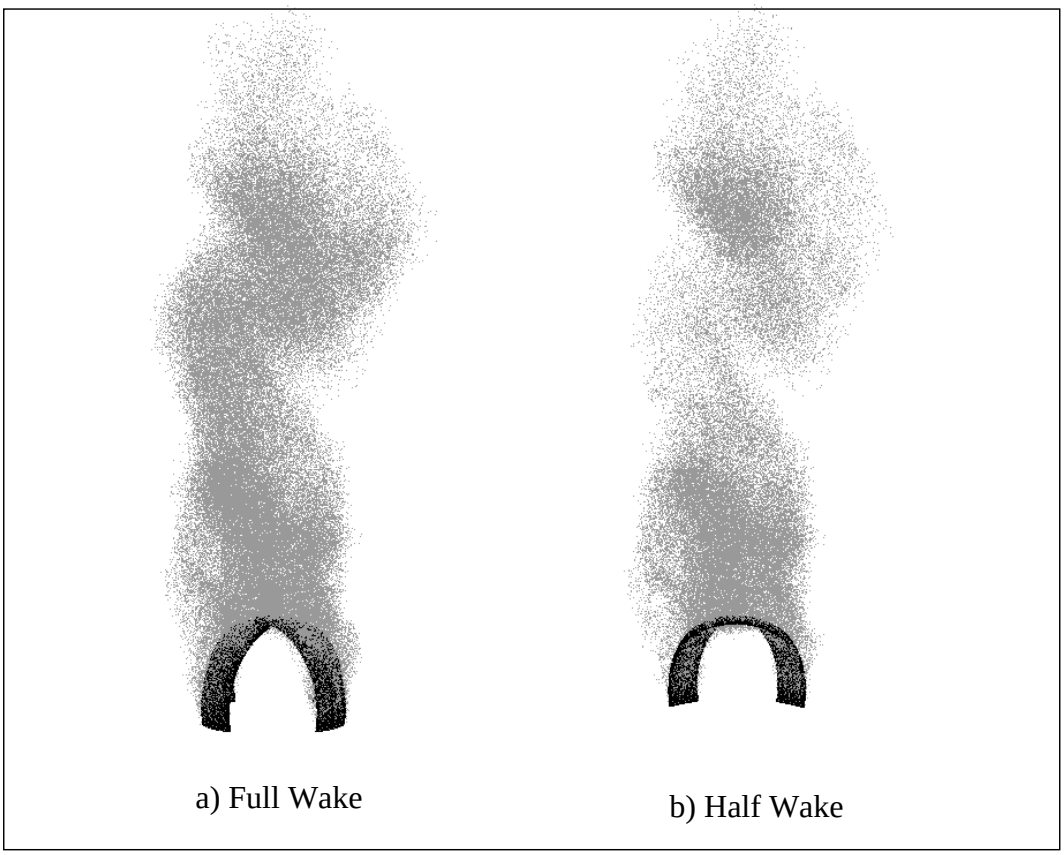


Figure 25. Wake Behind Rotating Cross Parachute

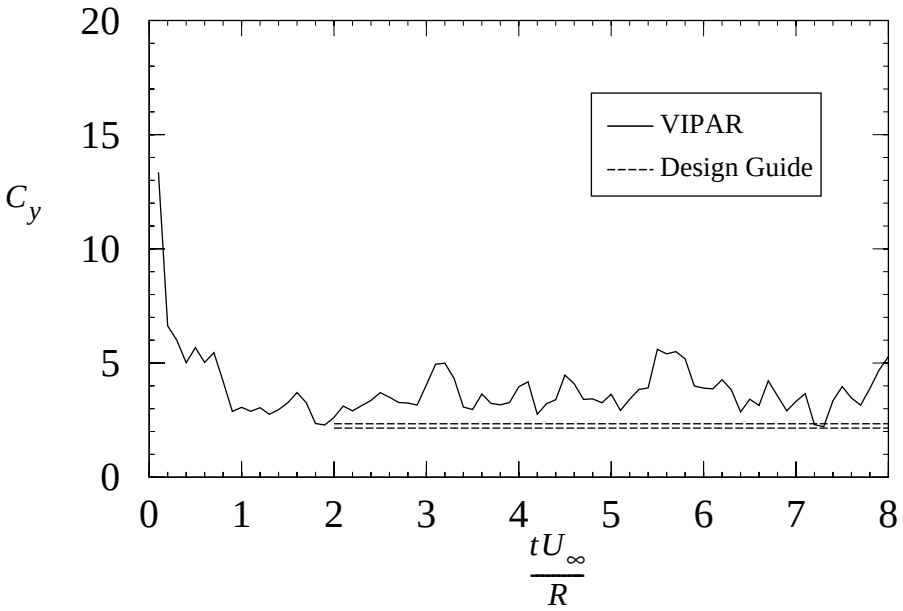


Figure 26. Axial Force Coefficient for Cross

9.1.3 Partial Sphere with Vent

Results are obtained for a partial sphere with a vent as shown in Figure 27. The vent is placed at a latitude of 72 degrees North while the skirt is at 9 degrees South. The surface was discretized into 624 triangular elements with 342 nodes.

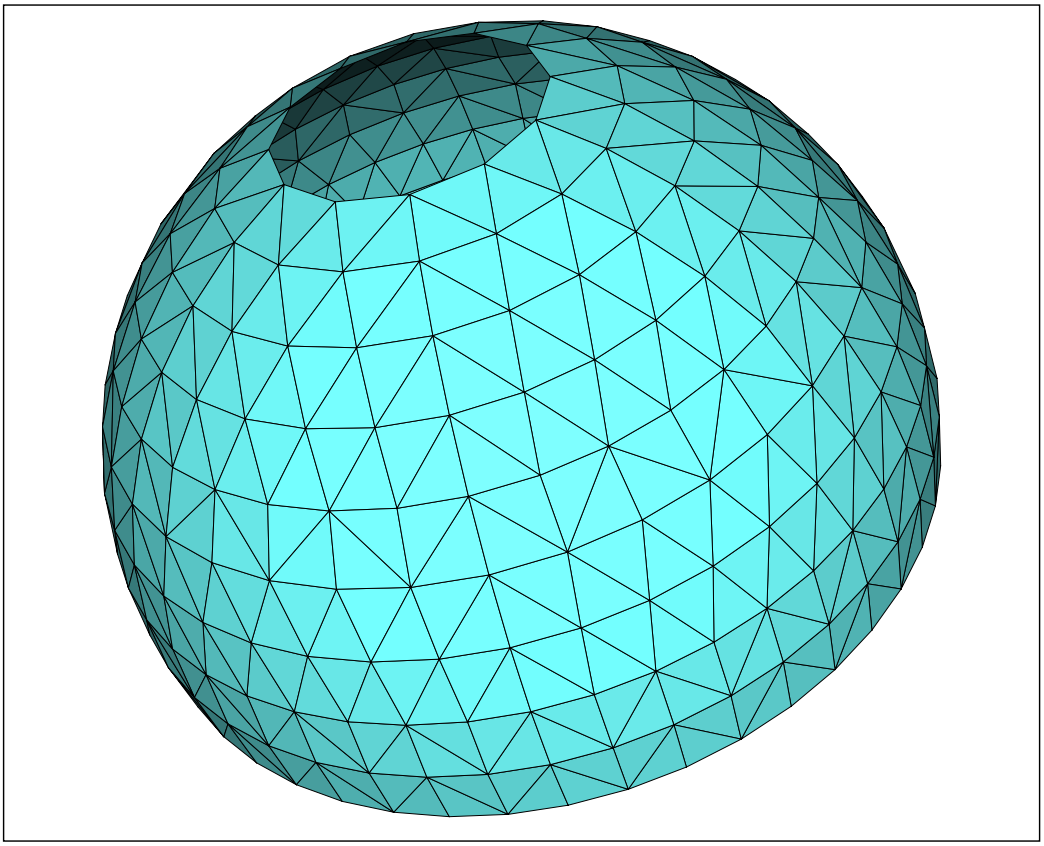


Figure 27. Surface Mesh for Partial Sphere

In Figure 28, the axial force coefficient time history is shown along with data given in the Recovery Systems Design Guide [31] for a parachute constructed from a flat circular plan form. C_{Do} design guide values range from 0.45 to 0.50 and S_o/R^2 in Equation 81 equals 10.79. The mean axial force coefficient represents the design guide data somewhat better. This geometry is better resolved than in the previous two simulations since the smallest feature of the solid parachute (the distance from vent to skirt) has twice the number of surface elements across it than does the smallest feature of the cross and ringslot (the distance across a ribbon). The axial drag coefficient for the partial sphere is very unsteady. The simulation results from VIPAR also show large unsteady lateral forces for this geometry that would give rise to the coning motion associated with solid canopies.

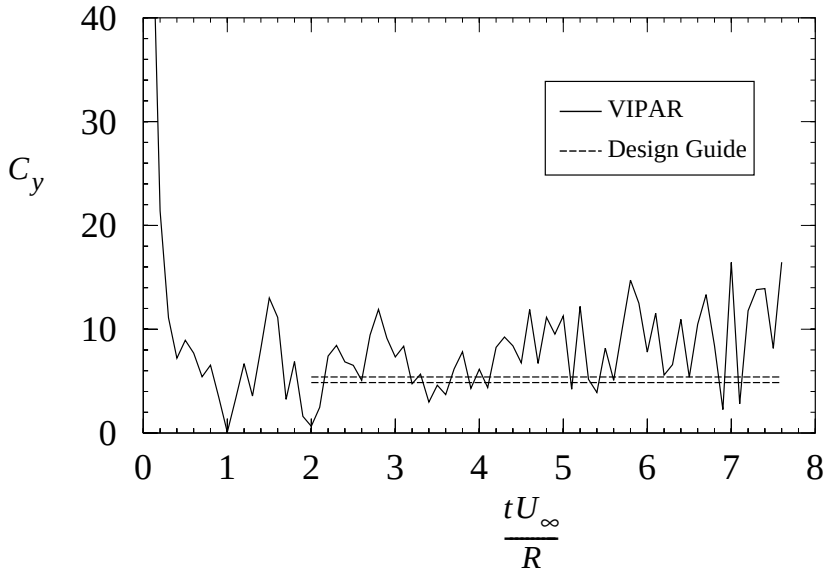


Figure 28. Axial Force Coefficient for Partial Sphere with Vent

In Figure 29, an initial puff through the vent in the form of a vortex ring is clearly seen. It is conjectured that these strong vortex rings play a vital role in the extreme fluctuations of the axial force coefficient. These fluctuations would give rise to the familiar “breathing” phenomenon associated with cloth parachutes.

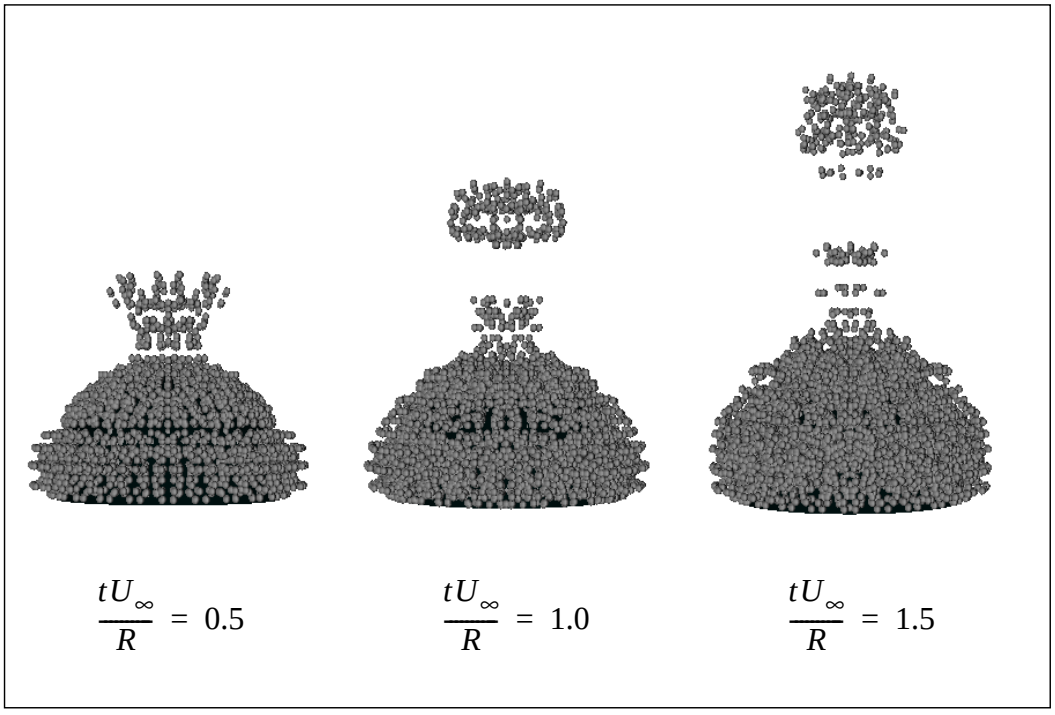


Figure 29. Developing Wake Behind a Partial Sphere with Vent

9.2 Flexible Geometries

In this section, we present preliminary results for simulations in which the VIPAR code is coupled with the structural mechanic code PRESTO. Both of these simulations were run on a SUN computer using four processors.

9.2.1 Impulsively Started Ribbons

In Figure 30, we show the wake development behind two flexible ribbons that are attached at their ends but are otherwise free to move. The ribbon ends are impulsively moved from rest to a uniform translational velocity. The end attachments are constrained to move in a direction normal to the ribbons in their initial positions. The two ribbons contain 400 triangular fluid elements and 200 quadrilateral structural shell elements. The simulation was run for 50 fluid time steps and 5260 structural time steps.

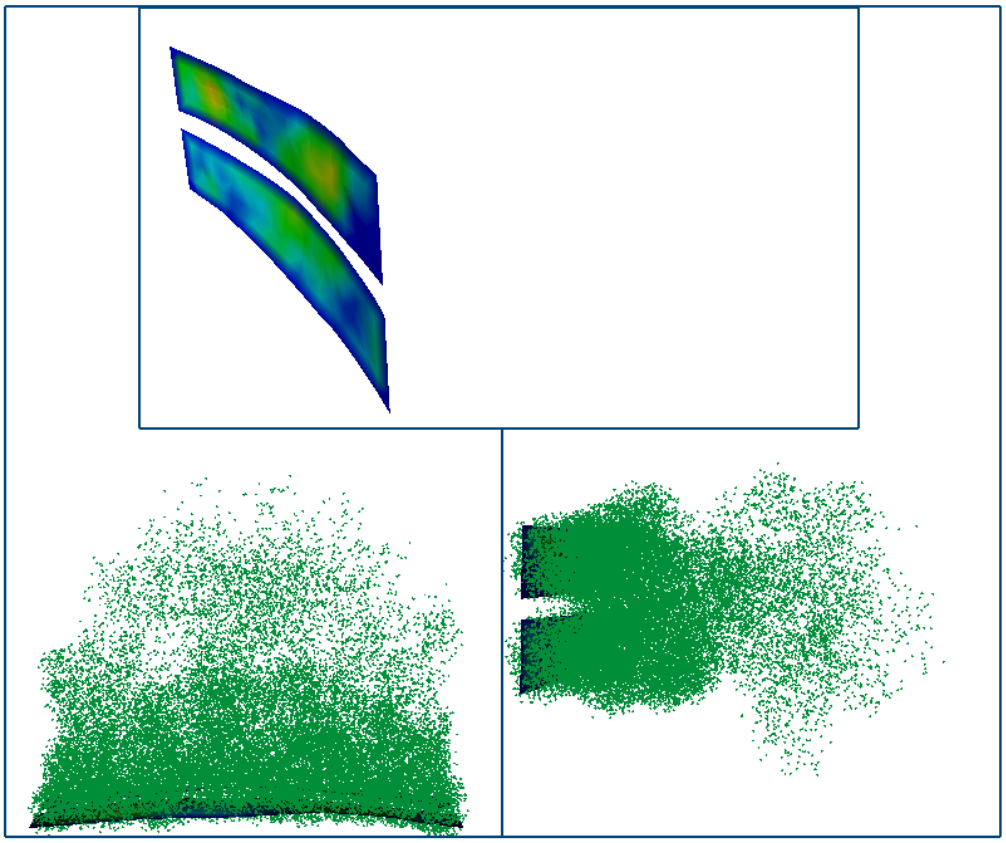
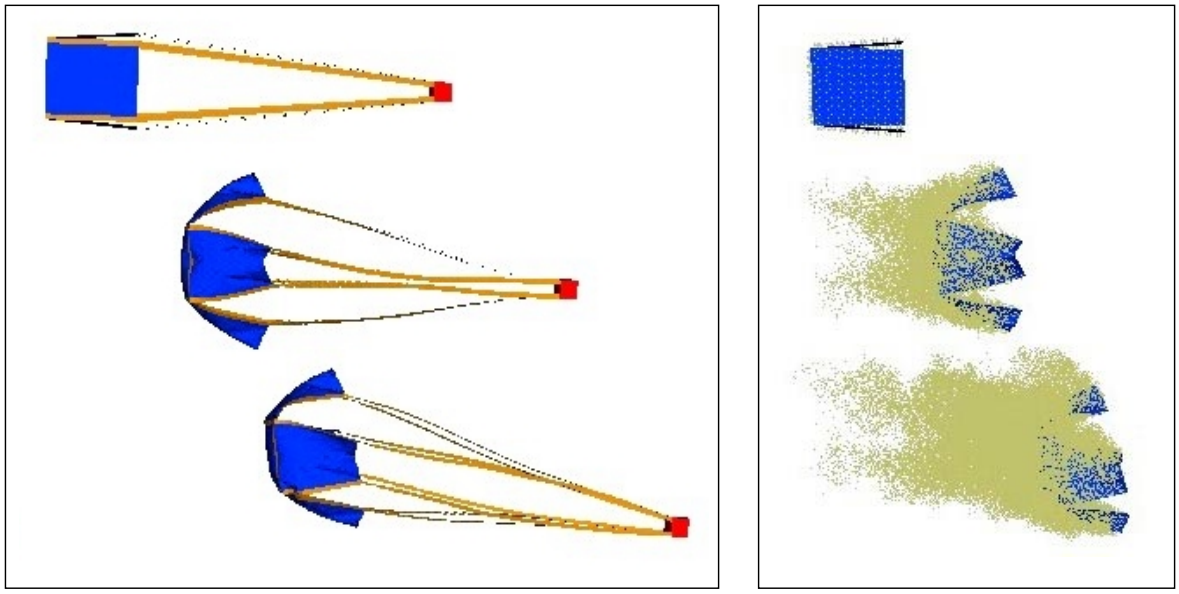


Figure 30. Impulsively Started Flexible Ribbons

9.2.2 Inflating Cross Parachute

In Figure 31, a fully coupled fluid-structure simulation for a cross parachute whose arm widths are 30% of the total arm lengths is shown. The initial shape of the cross parachute is that of an open-ended box. The fluid surface mesh consists of 840 triangular elements and 473 nodes. The structural model consists of 3616 quadrilateral shell elements with 4267 nodes. The parachute fabric itself is modeled with 1680 elements, the suspension lines (including the lines across the canopy) with 1552 elements, and the payload with 384 elements. The number of fluid time steps and structural time steps are 40 and 400 respectively. The parachute has an initial horizontal velocity and is impulsively inserted into quiescent air. After insertion, the parachute and its payload are allowed to decelerate and to begin to "turn over" under the influence of gravity. The behavior of this inflation simulation appears to be qualitatively correct.



a) PRESTO Output b) CURL Output

Figure 31. Inflating Cross Parachute

(this page intentionally left blank)

10 SUMMARY

VIPAR, a first generation fluid mechanics code using gridless vortex methods, has been developed and applied to impulsively started, fully inflated parachute shapes as well as a few other bluff bodies. The code has also been coupled with the structural dynamics code PRESTO within the Sierra framework. Several coupled calculations have been made to date.

Development of VIPAR is continuing and a number of algorithm enhancements are planned to extend code capabilities as well as to improve the accuracy and reduce CPU times. Examples include: the number of wake elements will be reduced significantly by shedding vorticity at the nodes rather than at the element centers; an improved stretching algorithm is planned to avoid unwanted loss of coherence due to diffusion in the wake; also, the extension of the gridless method into the compressible flow regime is underway [32].

There are several memory and data management issues which need to be addressed. While significant effort has already gone into minimizing memory requirements there are opportunities to reduce these further. Presently, the amount of data which is placed into the Exodus data base is large. Some of these data were originally generated to facilitate code debugging. In addition, it may be unnecessary to retain information at every computational time step. In short, the data management goal is to ensure that simulation results are efficiently captured.

A significant verification and validation (V&V) effort will be required in order to qualify and maintain the code. Preliminary V&V plans have been written. The evolution of the source code is actively tracked by the SIERRA Code Management System (SCMS). Regression testing of the code which exercises about 90% of the code is performed automatically each day. Verification tests will be added in which the code is required to simulate “manufactured solutions” which are analytical in nature. In addition, we will test the code against experimental data obtained from various bluff body flows such as those obtained by Higuchi [33] et al. for unsteady flow over disks.

(this page intentionally left blank)

11 REFERENCES

[1] Strickland, J. H., "Gridless Compressible Flow: A White Paper," Sandia National Laboratory Report SAND2001-0527, February, 2001.

[2] Information on EnSight™ software is available on the Web at: `http://www.ensight.com/default.html`

[3] Gharakhani, A., "A Regularized Galerkin Boundary Element Method (RGBEM) for Simulating Potential Flow About Zero Thickness Bodies," Sandia Laboratory Report, SAND99-2578, 1999.

[4] Gharakhani, A., "A 3D Galerkin Boundary Element Method for Potential Flow About Thin Membranes," 38th Aerospace Sciences Meeting & Exhibit, Reno NV, AIAA 2000-1000, January, 2000.

[5] Shankar, S. and van Dommelen, L., "A New Diffusion Procedure for Vortex Methods," *J. Comp. Phys.*, Vol. 127, pp. 88-109, 1996.

[6] Gharakhani, A., "A Higher Order Vorticity Redistribution Method for 3-D Diffusion in Free Space," Sandia National Laboratories Report SAND2000-2505, October 2000.

[7] Kane, T. R., *Dynamics*, Holt, Rinehart, and Winston, Inc., pp. 97-98, 113-115, 1968.

[8] Lamb, H., *Hydrodynamics*, Sixth Edition, p. 224, Dover Publications, New York, April 1932.

[9] Karamcheti, K, *Principles of Ideal-Fluid Aerodynamics*, John Wiley and Sons, New York, p. 530, 1966.

[10] Strickland, J. H., Webster, B. T., Nguyen, T., "A Vortex Model of the Darrieus Turbine: An Analytical and Experimental Study, Sandia Laboratory Report, SAND79-7058, p. 15, February, 1980.

[11] Ashley, H. and Landahl, M. T., *Aerodynamics of Wings and Bodies*, Addison-Wesley, Reading MA, pp. 44-45, 1965.

[12] White, F. M., *Fluid Mechanics*, McGraw-Hill Book Company, New York, p. 482, 1979.

[13] Strickland, J., "Axisymmetric Bluff-Body Flow: A Vortex Solver for Thin Shells," Sandia Laboratory Report, SAND91-2760, p. 54, May, 1992.

[14] Gharakhani, A., "A Survey of Grid-Free Methods for the Simulation of 3-D Incompressible Flows in Bounded Domains," Sandia National Laboratory Report, SAND97-2256, September, 1997.

[15] Batchelor, G. K., *An Introduction to Fluid Dynamics*, Cambridge University Press, pp 131-132, 1992.

[16] Strickland, J. H., Baty, R. S., "A Three-Dimensional fast Solver for Arbitrary Vorton Distributions," Sandia National laboratory Report, SAND93-1641, May, 1994.

[17] Strickland, J. H., Gritzo, L. A., Baty, R. S., Homicz, G. F., Burns, S. P., "Fast Multipole Solvers for Three-Dimensional Radiation and Fluid Flow Problems," *ESAIM Proceedings*, Vol. 7, pp. 408-417, 1999.

[18] Greengard, L., “Fast Algorithms for Classical Physics,” *Science*, Vol. 265, pp. 909-914, August 1994.

[19] Strickland, J. H. and Baty, R. S., “A Pragmatic Overview of Fast Multipole Methods,” *Lectures in Applied Mathematics*, Vol. 32, pp. 807-830, 1996.

[20] Homicz, G. F. and Burns, S. P., “Parallel Performance of VIPAR,” Sandia Memo, September, 22, 1998.

[21] Cheng, H., Greengard, L., Rokhlin, V, “A Fast Multipole Algorithm in Three Dimensions,” *Journal of Computational Physics*, Vol. 155, pp. 468-498, 1999.

[22] Chapra, S. C., and Canale, R. P., *Numerical Methods For Engineers With Personal Computer Applications*, p. 526, McGraw-Hill, 1985.

[23] Schoof, L. A., and Yarberry, V. R., EXODUS II: A Finite Element Data Model, Sandia National Laboratories Report SAND-92-2137, September, 1994.

[24] Network Common Data Format (netCDF) software is freely available at the following Web site: `http://www.unidata.ucar.edu/packages/netcdf/`

[25] ASCI Red home page at: `http://www.sandia.gov/ASCI/Red/main.html`

[26] NEMESIS documentation is available on-line in PDF format at: `http://endo.sandia.gov/SEACAS/Documentation/Nemesis_Users_Guide.pdf`

[27] Message Passing Interface Forum, MPI: A Message-Passing Interface Standard, Version 1.1, June 12, 1995. Available on-line at: `http://www.mpi-forum.org/index.html`

[28] Taylor, L. M., Edwards, H. C., and Stewart, J. R., “Functional Requirements for Sierra Version 1.0 Beta,” SAND99-2587, October, 1999. Also, see Sierra Home Page at: `http://www.cfd.sandia.gov/sierra.html`.

[29] Taylor, L. M., and Flanagan, D. P., “PRONTO3D A Three-Dimensional Transient Solid Dynamics Program,” Sandia National Laboratory Report, SAND87-1912, March, 1989.

[30] Strickland, J. H., Homicz, G. F., Gossler, A. A., Porter, V. L., “On the Development of a Gridless Inflation Code for Parachute Simulations,” AIAA-2001-2000, *16th AIAA Aerodynamic Decelerator Systems Technology Conference*, Boston MA, May 21-24, 2001.

[31] Ewing, E. G., Bixby, H. W., and Knake, T. W., “Recovery System Design Guide, Air Force Flight Dynamics Laboratory Technical Report AFFDL-TR-78-151, December, 1978.

[32] Strickland. J. H., “Gridless Compressible Flow: A White Paper,” Sandia National Laboratories Report, SAND2001-0527, February, 2001.

[33] Higuchi, H., Balligand, H., and Strickland, J. H., “Numerical and Experimental Investigation of the Unsteady Axisymmetric Flow Over a Disk,” *Journal of Fluids and Structures*, Vol. 10, No. 7. pp. 705-719, October, 1996.

APPENDIX

A.1 VIPAR CALL TREE

In this section, the call tree for the stand alone VIPAR program is outlined in Figures 32 and 33. The call tree does not include calls to EXODUS II routines or MPI routines. Calls to subroutines containing elementary vector operations are also excluded from these figures but are contained in the VIPAR subroutine descriptions in Section A.2 on page 67.

As can be seen from Figure 32, the main program “VIPAR” is simply a driver which reads primary inputs, manages the time stepping and calls the body motion generator. For situations where the code is coupled with the structural code PRESTO, only the program “VIPAR_SUB” is used. A Sierra “region” called “CURL” provides a wrapper around “VIPAR_SUB” so that it can be invoked by a Sierra “procedure” called “PURL.” PURL, in turn, handles the time-step control and coordination between the PRESTO and CURL regions.

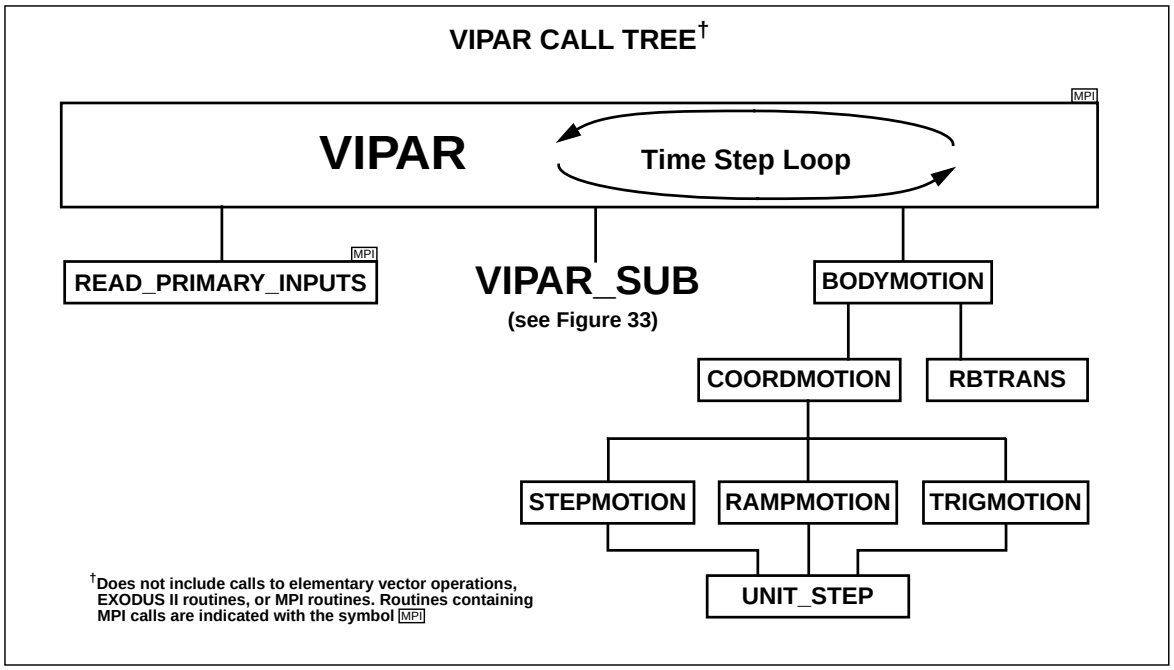


Figure 32. Schematic of VIPAR Call Tree

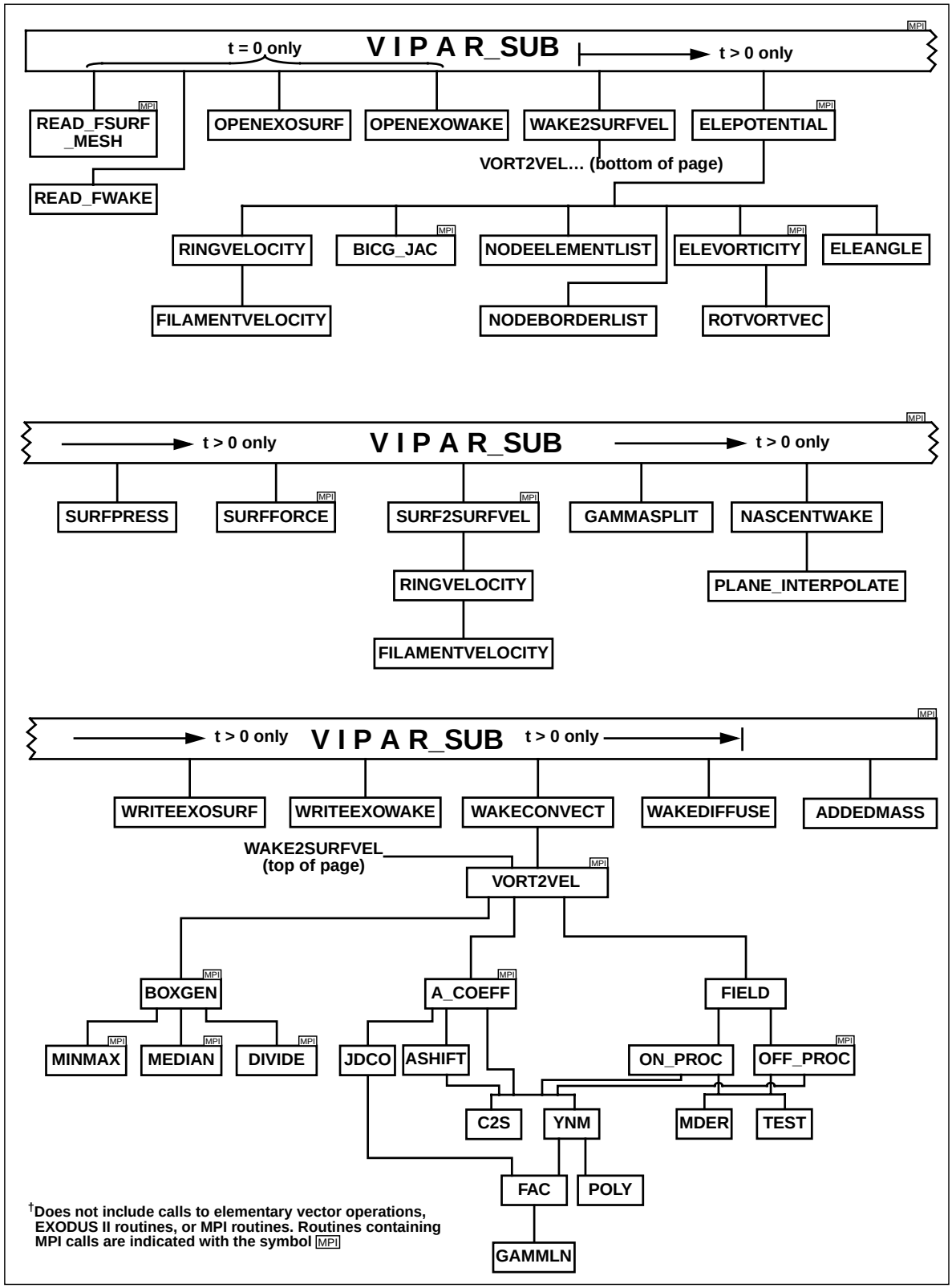


Figure 33. Schematic of VIPAR_SUB Call Tree

A.2 SUBROUTINE DESCRIPTIONS

This section contains a brief description of all VIPAR subroutines except those associated with calls to EXODUS II or MPI routines. More generous descriptions of each subroutine may be found in the prolog of each subroutine listing.

A_COEFF ..... Generates coefficients for multipole expansion in fast solver

ADDEDMASS ..... Calculates a diagonalized added mass matrix that yields element and nodal added mass vectors

ADDVECTORS..... Adds two vectors

ASHIFT ..... Shifts multipole coefficients to parent boxes for use in fast solver

ASOLVE ..... Required by subroutine linbcg

ATIMES..... Required by subroutine linbcg

BICG_JAC..... Bi-conjugate gradient linear solver

BODYMOTION ..... Moves the body from a position at time “t” to a position at time “t + delta_t”

BOXGEN..... Generates box structure for fast solver

CONSTTIMESVECTOR ..... Multiplies a constant times a vector

COORDMOTION ..... Computes the motion of the body fixed coordinate system

C2S ..... Transformation from Cartesian coordinates to spherical coordinates.

CROSSPRODUCT ..... Obtains the cross product of two vectors

DIVIDE ..... Splits parent box into its two children in fast solver

DOTPRODUCT..... Takes dot product of two vectors

ELEANGLE ..... Determines the included angle an element vertex

ELEPOTENTIAL ..... Computes potential jump on elements and nodes

ELEVORTICITY..... Obtains surface vortex sheet strength at elements and nodes

FAC ..... Calculates the factorial value of a number

FILAMENTVELOCITY..... Computes the velocity filed due to a vortex filament

FIELD ..... Computes vector potential and velocity fields from multipole and direct calculations

GAMMASPLIT..... Splits surface vortex sheet into two sheets, one on either side of the membrane

GAMLN	Computes the Gamma function
JDCO	Calculates the “ja” and “d” coefficient arrays for use in subroutine ASHIFT
MAGNITUDEVECTOR	Computes the vector magnitude
MDR	Calculates the partial derivatives of the potential function represented by the multipole expansion
MEDIAN	Determines the location of the median vortex element in a box
MINMAX	Determines the minimum and maximum coordinate values of the vortices in a box
NASCENTWAKE	Computes the nascent wake vorton strengths, locations, and sizes
NODEBORDERLIST	List of element nodes shared between processors
NODEELEMENTLIST	Computes the element numbers associated with each node
NORMALVECTOR	Produces a unit vector normal to two other vectors
OFF_PROC	Calculates the contribution to the potential and velocity fields at the target points held by a processor from the box multipole expansions and individual sources being held by other processors
ON_PROC	Calculates the contribution to the potential and velocity fields at the target points held by a processor from the box multipole expansions and individual sources being held by the processor
OPENEXOSURF	Opens the surface ExodusII data base
OPENEXOWAKE	Opens the wake ExodusII data base
PLANE_INTERPOLATE	Interpolates a scalar in a plane defined by three points with associated scalar values
POLY	Calculates the associated legendre polynomial
RAMPMOTION	Produces a ramp motion of the body
RBTRANS	Applies a rigid body translation and rotation
READ_FSURF_MESH	Reads the surface mesh ExodusII data base
READ_FWAKE	Reads a dummy wake file at first time step
READ_PRIMARY_INPUTS	Reads the input parameters for VIPAR
RINGVELOCITY	Computes the velocity field produced by a vortex ring at the boundaries of the triangular elements
ROTVORTVEC	Rotates linearly varying vorticity vectors on a triangular element so as to make them divergenc free

STEPMOTION	Produces a step motion of the body
SUBTRACTVECTORS	Subtracts one vector from another
SURF2SURFVEL	Computes the velocity produced on the surface elements and nodes by the surface vorticity itself
SURFORCE	Computes the integrated force vector due to the surface pressure distribution
SURFPRESS	Computes the surface pressure on the elements and nodes
TEST	Test for whether or not multipole expansions from a source box can be used at a target point
TRIGMOTION	Produces a trigonometric motion of the body
UNITVECTOR	Returns a unit vector in the direction of the input vector
UNIT_STEP	Produces a unit step
VIPAR_SUB	Gridless vortex fluids code
VORT2VEL	Computes the velocity field due to a vorticity field
WAKE2SURFVEL	Computes the velocity field at the surface due to the wake vortons
WAKECONVECT	Convects the wake elements
WAKEDIFFUSE	Diffuses the wake elements
WRITEEXOSURF	Writes the surface ExodusII data base
WRITEEXOWAKE	Writes the wake ExodusII data base
YNM	Calculates the value of the spherical harmonic
ZEROVECTOR	Returns a zero vector

A.3 NON-DIMENSIONAL VARIABLES

All variables in VIPAR are non-dimensional. An arbitrary reference length L_0 and reference velocity U_0 are used to accomplish this. The corresponding reference time T_0 is equal to L_0/U_0. The value L_0 will represent the size of the geometrical feature in the surface mesh whose dimension is unity. Likewise, the value U_0 will represent the velocity feature of the velocity time history whose magnitude is unity.

The following are typical non-dimensional variables denoted by an asterisk:

length l^* ..... $\frac{l}{L_0}$

velocity u^* ..... $\frac{u}{U_0}$

time t^* ..... $t\frac{U_0}{L_0}$

pressure p^* ..... $p\frac{\rho}{2g_c}U_0^2$, where ρ is the fluid density and where g_c is a constant that relates the units of force, mass, length and time. Several examples for g_c are:

$$g_c = 32.174\frac{\text{lb}_m}{\text{lb}_f}\frac{\text{ft}}{\text{sec}^2},\ 1.0\frac{\text{slug}}{\text{lb}_f}\frac{\text{ft}}{\text{sec}^2},\ 1.0\frac{\text{kg}}{\text{newton}}\frac{\text{m}}{\text{sec}^2}$$

force F^* ..... $F\frac{\rho}{2g_c}L_0^2U_0^2$

circulation Γ^* ..... $\frac{\Gamma}{L_0U_0}$

vortex sheet strength γ^* ..... $\frac{\gamma}{U_0}$

velocity potential ϕ^* ..... $\frac{\phi}{L_0U_0}$

kinematic viscosity ν^* ..... $\frac{\nu}{L_0U_0} = \frac{1}{R_e}$, where R_e is the Reynold's number based on L_0 and U_0

gradient operator ∇^* ..... $L_0\nabla$

A.4 INPUT FILES

In order to run VIPAR, one needs three types of input files. The first of these is a single "primary_inputs_file" which is entered on the command line at the time VIPAR.exe is executed. For example on Janus, VIPAR would be executed using a command line such as:

%yod -sz number_processors VIPAR.exe -i primary_inputs_file.

On the SUN, the command line would be of the form:

%mpirun -np number_processors VIPAR.exe -i primary_inputs_file.

Another file type (specified in the primary inputs file) that defines the initial surface geometry or mesh is also required. The number of surface mesh files will be equal to the number of processors. Lastly, a file type (also specified in the primary inputs file) that initializes the wake geometry is required. Here again, the number of files required will be equal to the number of processors.

A.4.1 The Primary Input File

The primary input file specifies time step information, the maximum number of vortex elements, the order of the multipole expansion, the Reynold's number, the rootnames for the input and output mesh and wake files, and information about the motion of the body. An example primary input file is given in Figure 34.

```
$PRIMARY_INPUTS
  time_step_beg   = 1
  time_step_end   = 10
  time_per_step   = 0.25D+0,

  max_num_vnod    = 3000
  nmax            = 2
  Re              = 1.0D+04
  mesh_rootname   ='my_geometry'
  wake_rootname   ='my_geometry'
  output_rootname ='my_geometry_out'

  motion_num      = 1,
  to              = 0.0D+0
  tf              = 12.5663706144D+0
  a1              = 0.0D+0, 0.0D+0, -4.0D+0
  b1              = 3*0.0D+0
 $END
```

Figure 34. Primary Input File

A.4.2 Input Mesh Files

PARTRAN™ is used to generate VIPAR surface mesh files which are output in Exodus data base format. The surface elements used in VIPAR are triangular shell elements. It is also

worth noting that one of the present authors (V. L. Porter) has written a C++ program to convert a surface made up of quadrilateral shell elements into one containing triangular shell elements with the same nodes. Appropriate geometries include only those that are made up of simply connected shell elements.

For parallel runs, it is first required that the Exodus surface mesh files be broken up into a series of files representing the portions of the surface which will reside on each processor. The easiest way to break up the Exodus mesh files for parallel VIPAR is to use the NEMESIS utility “loadbal.” For information on loadbal, enter loadbal -h at the command line prompt.

As an example, assume that one has a global mesh input file named my_geometry.mesh.exo.g and one wishes to run on 4 processors. The process for obtaining the four files is as follows:

1. Using your favorite editor, create an empty file:

   my_geometry.mesh.exo

2. On a command line with prompt % enter:

   % loadbal -p 2x2 my_geometry.mesh.exo

   The 2x2 indicates a 2x2 = 4 processor run.

   This creates 5 files:

   my_geometry.mesh.exo.cfg

   my_geometry.mesh.exo.lbd.err

   my_geometry.mesh.exo.lbd.out

   my_geometry.mesh.exo.nem

   my_geometry.mesh.exo.spd

3. On a command line enter:

   % my_geometry.mesh.exo.spd

   This creates a file named my_geometry.mesh.exo.pex and a subdirectory named 01.

   The four new Exodus files my_geometry.mesh.exo.par.p.0-4 are now in subdirectory 01.

A.4.3 Wake Input Files

While there are initially no wake elements, the structure of the Exodus data base requires that all of the wake elements be present from the very first time step. Therefore, the scheme is to create all of the wake elements and place them at the origin of the global coordinate system. This is accomplished by using a C++ program called “genwake” which is listed in Figure 35. For parallel vipar runs, the wake must also be divided among processors using loadbal as with the mesh input files.

Command line execution of genwake will produce the prompt “Enter the filename for the created wake mesh file:” such as my_geometry.wake.exo.g. The response to the second prompt “Enter a title for the database:” can be anything. The response to the third prompt “Enter the number of points you want created:” should be the total number of vortons wanted for the run (not the number of vortons per processor for the run).

Next proceed to break up the data into a set of files for each processor. Using your favorite editor, create an empty file named my_geometry.wake.exo.i. On a command line enter load-bal -p 2x2 my_geometry.wake.exo which creates 5 files:

my_geometry.wake.exo.cfg

my_geometry.wake.exo.lbd.err

my_geometry.wake.exo.lbd.out

my_geometry.wake.exo.nem

my_geometry.wake.exo.spd

Finally, enter my_geometry.wake.exo.spd on a command line which will create one file named my_geometry.mesh.exo.pex and one subdirectory named 01 in which there are new exodus files named my_geometrywake.exo.par.p.0-4.

```
// C++ program to generate a wake “mesh” file


#include <iostream>
#include <string>
#include <exodusII.h>

using namespace std;
int main()
{
 char title[MAX_LINE_LENGTH+1];
 string outfile;
 int num_pts = 0;
 cout << “Enter the filename for the created wake mesh file: “;
 cin >> outfile;
 cin.get();
 cout << endl;
 cout << “Enter a title for the database: “;
 cin.getline(title, MAX_LINE_LENGTH);
 cout << endl;
 cout << “Enter the number of points you want created: “;
 cin >> num_pts;
 cout << endl;
int exo_id;
 float ver;
 int CPU_ws = sizeof(float);
 int IO_ws_in = 0;
 int IO_ws_out = sizeof(float);
 int error;
// create exodus output file
 exo_id = ex_create(outfile.c_str(),EX_CLOBBER,&CPU_ws,&IO_ws_out);
```

Figure 35 Program GenWake (continued on next page)

```
// set initialization parameters
int num_dim = 3;
int num_elem_blk = 1;
int num_nsets = 0;
int num_ssets = 0;
int num_elem = num_pts;

error = ex_put_init(exo_id, title, num_dim, num_pts,
  num_elem, num_elem_blk, num_nsets, num_ssets);
// write qa info
int num_qa_records=1;
char* qa_record[1][4];

qa_record[0][0] = "GENWAKE";
qa_record[0][1] = "genwake";
qa_record[0][2] = "";
qa_record[0][3] = "";

error = ex_put_qa(exo_id, num_qa_records, qa_record);

// memory for element connectivity. Number of nodes per element is one.
int * connect = new int [num_elem];
for (int i = 0; i < num_pts; i++) connect[i] = i + 1;

// write out element block data and connectivity
char * elem_type = "SPHERE";
int num_nod_per_elem = 1;
int elem_block_id = 1;
int num_attr = 1;

error = ex_put_elem_block(exo_id, elem_block_id, elem_type,
     num_elem, num_nod_per_elem, num_attr);
error = ex_put_elem_conn(exo_id, elem_block_id, connect);

// flush the buffer
error = ex_update(exo_id);
delete [] connect;

cout << "Created wake file "<<outfile << "with "<< num_pts
<< "points. "<< endl

// close the file
error = ex_close(exo_id);

return 0;
}
```

Figure 35. Program GenWake (continued from previous page)

A.5 SURFACE OUTPUT FILES

A.5.1 Joining Surface Output Files

For parallel runs, an Exodus file containing computed variables is produced by each processor for its surface segment. Therefore, these files need to be joined in order to view the variables on the entire surface. This may be accomplished by use of the NEMESIS utility nem_join. By default, nem_join, reads a file nem_join.inp. The process for obtaining the concatenated Exodus file is to first place files to be joined in subdirectory 01 and then enter nem_join on a command line. This single output file my_geometry_join.surf.exo.par is produced as a result of this. An example nem_join.inp file is shown in Figure 36.

```
Input FEM file = my_geometry.mesh.exo
        Scalar Results FEM file = my_geometry_join.surf.exo.par
        Use Scalar Mesh File = no
        Parallel Results file base name = my_geometry_out.surf.exo.par
        Number of Processors = 4
        Debug = 8
        Parallel Disk Info = number = 1
        Parallel file location = root =./0, subdir =.
```

Figure 36. File nem_join.inp for Surface

A.5.2 Global Surface Data

number_of_nodes . . . . . total number of surface nodes

number_of_elements . . . . . total number of triangular shell elements

cx, y, z . . . . . integrated surface pressure (force) coefficients

A.5.3 Nodal Surface Data

fnodx, y, z . . . . . nodal coordinates

fnodunode, vnode, wnode . . . . . nodal velocities

fnoduwake, vwake, wwake . . . . . velocities at nodes from wake vortons

fnodpotent . . . . . velocity potential at nodes

fnodnvx, y, z. . . . . surface normal vector at nodes

fnodgamma . . . . . vortex sheet strength at nodes

fnodgammauvx, y, z. . . . . unit vector in the direction of vorticity at the node

fnodpress . . . . . pressure jump across the membrane at the node

A.5.4 Element Surface Data

felex, y, z . . . . . element center coordinates

felenvx, y, z . . . . . element surface normal vector $\hat{n}$

felepotent. . . . . average velocity potential on element

feletaux, y, z . . . . . unit vector $\hat{\tau}$ along element edge

feleetax, y, z . . . . . unit vector $\hat{\eta}$ in element plane $\hat{\tau} \times \hat{\eta} = \hat{n}$

felegamma. . . . . average vortex sheet strength on element

felegammuvx, y, z . . . . . unit vector in the direction of vorticity on element

felegamrot . . . . . rotation angle which would be required to make the piecewise linear vorticity solenoidal (Note: this quantity is never used)

feleb2bvelu, v, w . . . . . velocity induced on an element by the body

felewakevelu, v, w . . . . . velocity induced on an element by the wake

felegamuvx, y, z . . . . . unit vector in the direction of element vorticity

felepress . . . . . pressure jump across the element

A.6 WAKE OUTPUT FILES

A.6.1 Joining Wake Output Files

As with the surface mesh files, an Exodus file containing computed variables is produced by each processor for its surface segment. Therefore, these files need to be joined in order to view the variables on the entire surface. This may again be accomplished by use of the NEMESIS utility nem_join. By default, nem_join, reads a file nem_join.inp. The process for obtaining the concatenated Exodus file is to first place files to be joined in subdirectory 01 and then enter nem_join on a command line. This single output file my_geometry_join.wake.exo.par is produced as a result of this. An example nem_join.inp file is shown in Figure 37.

```
Input FEM file = my_geometry.wake.exo
        Scalar Results FEM file = my_geometry_join.wake.exo.par
        Use Scalar Mesh File = no
        Parallel Results file base name = my_geometry_out.wake.exo.par
        Number of Processors = 4
        Debug = 8
        Parallel Disk Info = number = 1
        Parallel file location = root =./0, subdir =.
```

Figure 37. File nem_join.inp for Wake

A.6.2 Nodal Wake Data

vor_x, y, z. . . . . coordinates of vorton center

vor_lx, y, z . . . . . vorton stick vector

vor_vhx, y, z. . . . . coordinates of vorton head (head of stick vector)

vor_vtx, y, z . . . . . coordinates of vorton tail (tail of stick vector)

vor_gam . . . . . vorton circulation

A.6.3 Element Wake Data

radius . . . . . core radius of vorton element

status . . . . . status is equal to 0 for an inactive vorton (pre-nascent) or 1 for an active vorton

A.7 Average Velocity on an Element Due to a Point Vorton.

The velocity $\vec{u}$ is given in terms of the vector potential $\vec{\Phi}$ as:

$$\vec{u} = \nabla \times \vec{\Phi}. \tag{82}$$

For a point vorton, the vector potential is given by:

$$\vec{\Phi} = \frac{\Gamma \Delta s}{4\pi r} \vec{e_v}, \tag{83}$$

where $\Gamma \Delta s$ is the strength of the vorton and $\vec{e_v}$ is its direction. The velocity in the direction of the normal $\vec{n}$ to a surface A is given by:

$$u_n = \hat{n} \cdot (\nabla \times \vec{\Phi}), \tag{84}$$

and the velocity vector which is tangent to the surface containing the unit vectors $\hat{i}$ and $\hat{j}$ is given by:

$$\vec{u}_\tau = (\hat{i} + \hat{j}) \cdot (\nabla \times \vec{\Phi}) \tag{85}$$

Averaging the velocity over the surface area A gives:

$$\vec{\bar{u}}_A = \frac{1}{A} \int_A (\nabla \times \vec{\Phi}) dA. \tag{86}$$

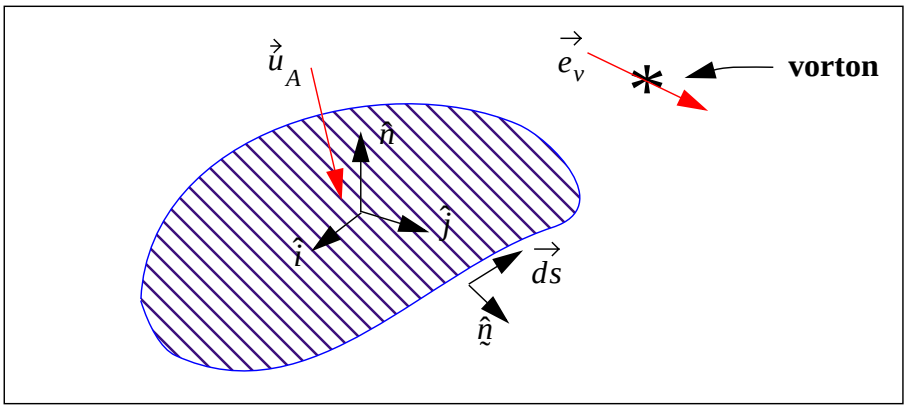


Figure 38. Influence of a Vorton on a Surface Area

The normal component of the area averaged velocity may be obtained from Stokes' Theorem:

$$\bar{u}_n = \frac{1}{A} \int_A (\nabla \times \vec{\Phi}) \cdot \hat{n} dA = \frac{1}{A} \oint_s \vec{\Phi} \cdot \vec{ds} = \frac{\Gamma \Delta s}{4\pi A} \oint_s \frac{1}{r} (\vec{e_v} \cdot \vec{ds}). \tag{87}$$

If the area is described by a polygon, the contour integration may be constructed by considering each side separately as indicated in Figure 39.

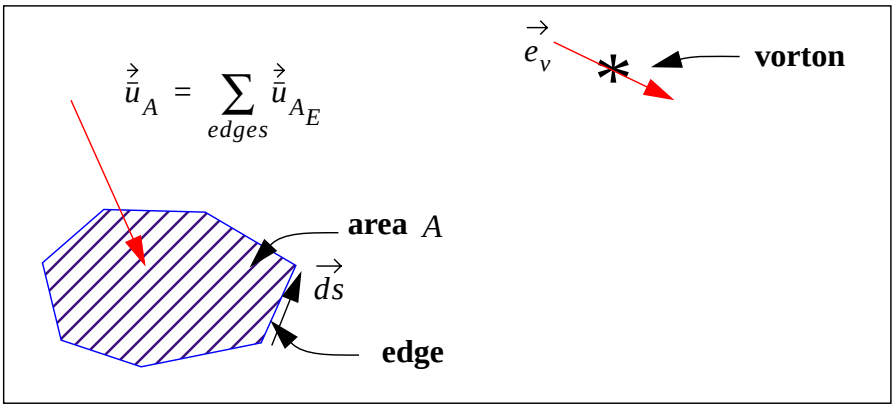


Figure 39. Use of edge data to compute area averaged velocity

In considering the contribution from a single edge, we start with a coordinate system which is in a plane defined by the edge and the vorton. The coordinate system is aligned with the edge as shown in Figure 40

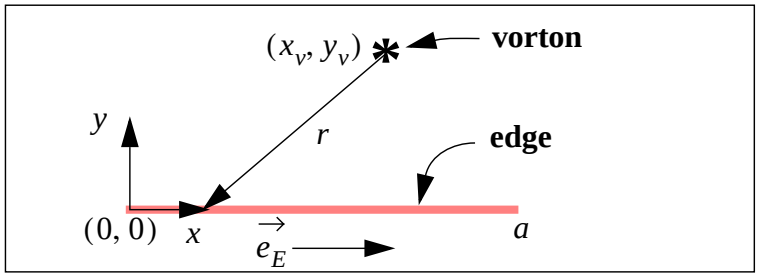


Figure 40. Edge Coordinate System

The contribution to $\bar{u}_n$ from the edge shown in Figure 40 is given by:

$$\bar{u}_{nE} = \frac{\Gamma \Delta s}{4\pi A}\int_0^a \frac{1}{r}\vec{e}_v \cdot \vec{dx} = (\vec{e}_v \cdot \vec{e}_E)\frac{\Gamma \Delta s}{4\pi A}\int_0^a \frac{1}{\sqrt{(x_v - x)^2 + y_v^2}}dx\,. \tag{88}$$

Integrating Equation 88 yields:

$$\bar{u}_{nE} = \frac{\Gamma \Delta s}{4\pi A}\ln\left(\frac{x_v + \sqrt{x_v^2 + y_v^2}}{x_v - a + \sqrt{(x_v - a)^2 + y_v^2}}\right)(\vec{e}_v \cdot \vec{e}_E)\,. \tag{89}$$

This argument of the logarithm may also be written in terms of the length of the sides (shown in Figure 41) of the triangle whose vertices are at $(0, 0)$, $(a, 0)$, and (x_v, y_v) in Figure 41.

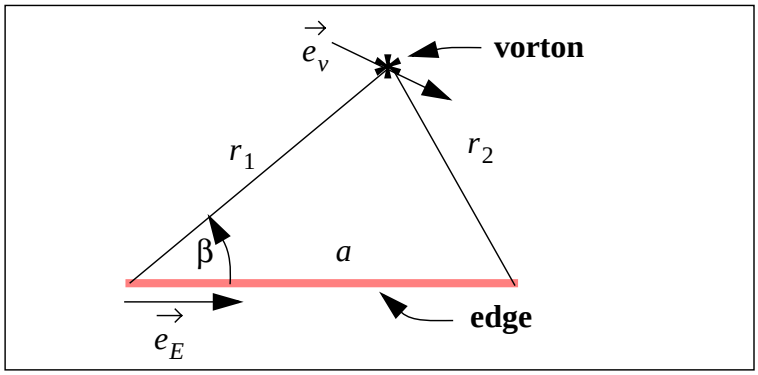


Figure 41. Edge Vorton Triangle

The two sides r_1 and r_2 are given by:

$$r_1 = \sqrt{x_v^2 + y_v^2}, \tag{90}$$

$$r_2 = \sqrt{(x_v - a)^2 + y_v^2},$$

which appears in the numerator and denominator of the log argument of Equation 89. Using the triangle identities, x_v may be written as:

$$x_v = r_1 \cos\beta = \frac{a^2 + r_1^2 - r_2^2}{2a}. \tag{91}$$

Substituting Equations 90 and 91 into 89 yields:

$$\bar{u}_{nE} = \frac{\Gamma \Delta s}{4\pi A} \ln\left[\frac{(r_1 + a)^2 - r_2^2}{-(r_2 - a)^2 + r_1^2}\right](\vec{e}_v \cdot \vec{e}_E). \tag{92}$$

A.8 Potential at a Point Due to a Vortex Filament.

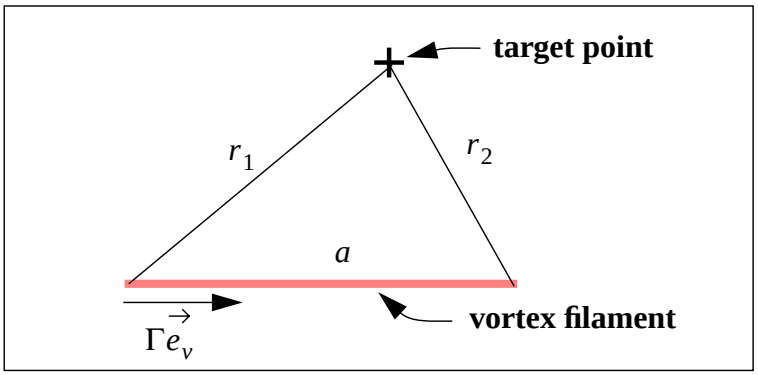


Figure 42. Influence of a Vortex Filament on a Target Point

The influence of a vortex filament on a single target point can be written as:

$$\vec{\Phi} = \int_0^a \frac{\Gamma}{4\pi r} \vec{e}_v dx . \tag{93}$$

By analogy with the previous section (Equation 88 and 92) we see that:

$$\vec{\Phi} = \frac{\Gamma}{4\pi} \ln \left[\frac{(r_1 + a)^2 - r_2^2}{-(r_2 - a)^2 + r_1^2} \right] \vec{e}_v . \tag{94}$$

(this page intentionally left blank)

Distribution

2 Analytical Methods (2)
2133 152nd Ave.
Redmond, WA 98052
Attn: F. Dvorak
B. Maskew

1 Dr. P. W. Bearman
Dept. of Aeronautics
Imperial College
London, SW7 2BY
ENGLAND

1 Dr. J. Carrier
124 Me Pierre Termier
73000 Chambery, FRANCE

1 Prof. A. J. Chorin
Department of Mathematics
University of California
Berkeley, CA 94720

1 Prof. Tim Colonius
Mechanical Engineering
Mail Code 104-44
California Institute of Technology
Pasadena, CA 91125

1 Dr. Luca Cortelezzi
Department of Mech. Eng.
McGill University
817 Sherbrooke Street West
Montreal, Quebec, H3A 2K6

1 Dr. R. Cortez
Dept. of Mathematics
Tulane University
6823 St. Charles Ave, #424
New Orleans, LA 70118

1 Dr. Jeff Eldridge
Department of Engineering
Trumpington Street
Cambridge University
Cambridge Cb4 1HH, UK

1 Dr. Adrin Gharakhani
Applied Scientific Research
1800 E. Garry Ave., Suite 214
Santa Ana, CA 92705

1 Prof. Ahmed F. Ghoniem
Massachusetts Institute of Technology
Mechanical Engineering Dept.
Room 3-342
Cambridge, MA 02139-4307

5 Prof. Albert A. Gossler
Mechanical Engineering Dept.
Northern Arizona University
Flagstaff, AZ 86011

1 Dr. John R. Grant
Naval Undersea Center
1176 Howell Street
Building 108, Code 8233
Newport RI 02841-1708

1 Prof. Leslie Greengard
Courant Institute of Math. Sciences
New York University
251 Mercer Street
New York, NY 10012

1 Prof. C. E. Hailey
Mech.anical Eng. Dept.
University Of Texas at Tyler
3900 University Blvd.
Tyler, TX 75799

1 Prof. H. Higuchi
Dept. of Mech. & Aero. Eng.
Syracuse University
Syracuse, NY 13244

1 Dr. Stephen Huyer
Naval Undersea Weapon Center
Hydrodynamics Branch, Code 8233
Building 1302/1
Newport, RI 02841

1 Dr. Vinay Kalro
Fluent, Inc.
1508 Oak Ave., #3N
Evanston, IL 60201

1 Prof. A. R. Karagozian
UCLA
46-147D Engr. IV
Mail Code 951597
Los Angeles, CA 90095-1597

1 Prof. Joseph Katz
Dept. Aerospace Eng. and Eng. Mech.
San Diego State University
San Diego, CA 92182-0183

1 Prof. Omar M. Knio
Dept. of Mech. Eng.
Johns Hopkins University
Baltimore, MD 21218-2686

1 Prof. Anthony Leonard
Graduate Aeronautics Lab.
1200 East California Blvd.
California Institute of Technology
Pasadena, CA 91125

1 Prof. S. Mas-Gallic
Centre de Mathematiques Appliquees
Ecole Polytechnique
91128 Palaiseau Cedex FRANCE

1 Dr. Jeffery S. Marshall
Iowa Institute of Hydraulic Research
University of Iowa
300 S. Riverside Drive
Iowa City, IA 52242-1585

1 Dr. Eckart Meiburg
Dept. of Aerospace Eng.
University of Southern California
854 W. 36th Place
Los Angeles, CA 90089-1191

1 Dr. D. I. Meiron
Dept. of Applied Mathematics
California Institute of Technology
Pasadena, CA 91125

1 Dr. M. Nitsche
Dept. of Math. & Stat.
University of New Mexico
Albuquerque, NM 87131

1 Prof. Y. Ogami
Dept. Mech. Eng.
Ritsumeikan University
Kusatsu 525-77, JAPAN

1 Prof. Ion Paraschivoiu
Dept. of Mech. Eng.
Ecole Polytecnique
CP 6079
Succursale A
Montreal H3C 3A7, CANADA

1 Prof. S. C. Rajan
Dept. of Aerospace Eng.
Indian Institute of Technology, Madras
Chennai, Pin 600036
Tamilnadu, India

1 Prof. V. Rokhlin
Department of Computer Science
Yale University
PO Box 2158
New Haven, CT 06520

1 Prof. P. G. Saffman
Dept. of Applied Mathematics
California Institute of Technology
Pasadena, CA 91125

1 Prof. William Saric
Dept of Mech. and Aerospace Eng.
Arizona State University
PO Box 876106
Tempe, AZ 85287-6106

1 Prof. T. Sarpkaya
Dept. Mech. Eng.
Code 69-SL
Naval Postgraduate Academy
Monterey, CA 93943

1 Prof. Michael S. Selig
Dept. of Aero. and Astro. Engineering
University of Illinois
306 Talbot Lab
Urbana, IL 61801

1 Prof. J. A. Sethian
Dept. of Mathematics
University of California
Berkely, CA 94720

1 Dr. David Sharpe
Department of Aeronautical Eng.
Queen Mary College
Mile End Road
London, E1 4NS
ENGLAND

1 Prof. Roger L. Simpson
Dept. Aerospace and Ocean Eng.
Virginia Polytechnic Institute
and State University
Blacksburg, VA 24061

1 Dr. Marios C. Soteriou
UTRC
411 Silver Lane
Mail Stop 129-15
East Hartford, CT 06108

1 Prof. John A. Strain
Department of Mathematics
University of California
Berkeley, CA 94720

1 Dr. S. Shankar
Quantum Corporation
MS E23
333 South Street
Shrewsbury, MA 01545

1 Dr. J. W. Oler
Texas Tech University
Dept. of Mech. Eng.
Lubbock, TX 79409

1 Prof. G. Trygvasson
Dept. Mech. Eng. and App. Mech.
University of Michigan
Ann Arbor, MI 48109-2125

1 Prof. O. R. Tutty
Dept. of Aero. & Astro.
University of Southampton
Highfield, Southampton, SO17 1BJ
Hampshire, United Kingdom

2 University of New Mexico
Dept. of Mech. Eng.
Albuquerque, NM 87106
Attn: M. S. Ingber
C. R. Truman

1 Dr. James Uhlman
Antheon Corporation
1 Corporate Place
Middletown, RI 02842-6277

2 U.S. Army Soldier & Biological
Chemical Command
Soldier Systems Center Natick
Kansas Street
Natick, MA 01760-5017
Attn: SSCNC-UTS (Richard Benney)
SSCNC-UTS (Keith Stein)

1 Prof. L. Van Dommelen
FAMU-FSU College of Engineering
2525 Pottsdamer Street, Room 229
Florida State University
Tallahassee, FL 32310-6046

1 Prof. C. W. Van Atta
Dept. of App. Mech. and Eng. Sci.
Mail Code 0411
University of California
La Jolla, CA 92093-0411

3 Washington State University (3)
Dept. Mech. & Matl's Eng.
Pullman, WA 99164-2920
Attn: Jacob Chung
Clayton Crowe
Tim Troutt

1 Prof. R. E. Wilson
Dept. Mech. Eng.
Oregon State University
Corvallis, OR 97331

1 Prof. G. S. Wincklemans
Center for Sys. Eng. & App. Mech.
Unité TERM, place du Levant 2,
Université Catholique de Louvain
Louvain-la-Neuve 1348
BELGIUM

1 Prof. Norman J. Zabusky
Dept. of Mech. and Aerospace Eng.
Rutgers University
PO Box 909
Piscataway, NJ 08855-0909

1 Dr. David Zielke
Naval Research Lab, 6-4-40
4555 Overlook Avenue, SW
Washington, DC 20375

1 0819 S. P. Burns, 9231
1 0819 T. E.Voth, 9231
1 0824 A. C. Ratzel, 9110
1 0825 F. G. Blottner, 9115
1 0825 W. H. Rutledge, 9115
1 0825 W. P. Wolfe, 9115
1 0826 W. L. Hermina, 9113
1 0834 J. E. Johannes, 9114
1 0834 M. R. Prairie, 9112
5 0835 G. F. Homicz, 9141
1 0835 S. N. Kempka, 9141
1 0835 M. J. McGlaun, 9140
1 0835 J. S. Peery, 9142
10 0835 J. H. Strickland, 9141
1 0836 R. O. Griffith, 9117
1 0836 E. S. Hertel, 9116
1 0836 P. E. DesJardin, 9132
1 0836 C. W. Peterson, 9100
1 0836 S. R. Tieszen, 9132
1 0841 T. C. Bickel, 9100
5 0847 V. L. Porter, 9142
1 1135 L. A. Gritzo, 9132
1 9051 H. N. Najm, 8351
1 9051 L. A. Rahn, 8351
1 9051 W. T. Ashurst, 8351

1 9018 Central Technical Files, 8945-1
2 0899 Technical Library, 9616
1 0612 Review & Approval Desk,
9612 For DOE/OSTI