Forces on underwater vehicles

by

Søren Trøjgård Jensen

Submitted to the Department of Ocean Engineering
in partial fulfillment of the requirements for the
degree of
Master of Science in Ocean Engineering
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
January 1995

Author. . . . .
Department of Ocean Engineering
Januarv 20th 1995

Certified by . . . .
Professor Jerome H. Milgram
Thesis Supervisor

Accepted by . . . .
Professor A. Douglas Carmichael
Chairman. Graduate Thesis Committee

Forces on underwater vehicles

by
Søren Trøjgård Jensen

Submitted to the Department of Ocean Engineering
in partial fulfillment of the requirements for the
degree of
Master of Science in Ocean Engineering

Abstract

A Boundary Element Integral Method code was developed to calculate the forces and moments on an underwater vehicle operating near the sea floor using Green's Theorem. The code is a potential based Boundary Element Integral Method code which models the body with quadrilateral planar panels using a constant source and dipole distribution and solves for the perturbation potential. The perturbation velocities are found by numerical differentiation of the perturbation potential by fitting a quadratic polynomial to the potential at three consecutive panels. The forces and moments are found by pressure integration. The code can in theory solve for arbitrary body shapes, but the present form of its panelizer accepts only axisymmetric bodies and bodies with elliptical cross sections. Calculations were done on some special geometries and their results compared to potential flow theory and to slender body theory.

Two experiments were conducted at the MIT Marine Hydrodynamics Water Tunnel to investigate the hydrodynamic forces on an underwater vehicle operating near a wall. In the experiments the vehicle distance from the wall and its pitch were varied. The measured results show that when the vehicle is pitched then flow separation have significant effect on the hydrodynamic forces.

Thesis Supervisor: Jerome H. Milgram
Title: Professor of Ocean Engineering

List of figures

List of tables

Chapter 1

Introduction

1.1 Background

This thesis is part of a larger research effort to develop tools for evaluation, design and control of Autonomous Underwater Vehicles (AUV). The emphases in this project are on investigating the hydrodynamic effects on an underwater vehicle operating near a solid boundary such as the sea bottom or a host ship.

Underwater vehicles serve important purposes in today's world. Besides the widespread military use, underwater vehicles perform many other tasks such as maintenance and damage inspections of offshore platforms and as reconnaissance vehicles searching for shipwrecks or other lost objects. Manned submersibles have been widely used ever since their introduction during the civil war, but their use is restricted by the size requirements and the needed safety precautions due to their human "cargo". Unmanned underwater vehicles have a wide variety of advantages over manned vehicles. First, the room for the operator is not needed so the vehicles can be smaller. This enables the vehicles to operate in confined spaces such as caves and shipwrecks. Second, the consequences from loss of the unmanned vehicle do not have the same catastrophic dimensions as with manned vehicles. Another practical issue is that the smaller unmanned vehicles will need smaller and less powerful thrusters for maneuvering. Smaller thrusters are less likely to "kick up" fine sediments from the sea floor that can impair visibility significantly. For the reasons mentioned, a large effort has been invested in developing the smaller lower-cost unmanned underwater vehicles.

In Remotely Operated underwater Vehicles (ROV) the human operator is positioned on a host surface ship or on another submersible nearby. The ROV can be fitted with many devices ranging from video monitors to multiple robotic arms to operate equipment and retrieve objects. The operation of an ROV is restricted due to the fact that it requires a cable connection to its host ship. The cable providing power and communication is dragged by the ROV. In some situations the ROV can still be more versatile than the manned underwater vehicle mainly due to the smaller size plus the advantages stated above. Miniature ROV's have proven very useful in harsh environments such as nuclear power plants and sewage pipelines, where they provide the platform for video cameras for "live" inspections.

With rapid development in the fields of electronics and control systems during the last two decades a base has been established for the first generation of fully Autonomous Underwater Vehicles (AUV). These vehicles are capable of following a preprogrammed course while collecting oceanographic data or performing other simple tasks. The next generation of AUV is expected to have enough computer power to autonomously correct for changes in the immediate surrounding environment e.g., avoid unexpected obstructions, correct for currents etc. without loosing the overall objective of the mission. The goal is to develop AUVs that are capable of undertaking long range missions lasting several weeks or maybe months. This may cut cost significantly due to the reduced need for a host ship.

Two important issues for development of the next generation of AUV's are:

- power storage for propulsion and computer equipment onboard AUV
- algorithms for the control system, especially while operating near a boundary

Batteries which provide silent operation, easy power control and simple operation have been the obvious choice of power source for AUV's. Lately other alternatives have been available like proton-exchange-membrane (PEM) fuel cells. This technology uses hydrogen and oxygen to create electric power with only water and heat as byproducts. PEM fuel cells do not have the long turn around time for recharging or the limited range as batteries do and they are very promising as the future power source for AUV's.

The pursuit for creating tools enabling the naval architect to predict the hydrodynamic effects on the submersible vehicle and solving the control problem has been on-going ever since humans have been able to navigate beneath the ocean surface. The control problem is of much greater interest when the underwater vehicle operates near a boundary such as the host ship or the sea bottom where the consequences from differences between predictions and real life effects can prove disastrous.

Different mathematical models have been developed to calculate the hydrodynamic effects on submerged vehicles. A usual approach for interaction effects is the invicid potential flow model which neglects the viscous effect of the fluid. The combination of a source and a sink has long been used to construct geometrical simple bodies such as spheres and ovoids. A derivative of this method is the sink and source line. By defining a continuos distribution of sink and sources along the centerline of the vehicle, combined with a uniform incoming flow, it is possible to model smooth axisymmetric bodies and calculate pressure and velocity distributions on the body. Not all objects in a flow are axisymmetrical, so the boundary integral element method (BIEM) for non-lifting bodies, Hess and Smith [2], was developed to handle arbitrary body shapes. The Hess and Smith approach has been widely applied with different modifications such as variable source strength and a combined source and dipole strength, Newman[8]. Methods for lifting bodies have been developed as well (Hess [7] and Morini and Kou [8]) using a combined source and dipole (vorticity) distribution and applying the Kutta condition e.g., no jump in velocity (or pressure) at the trailing edge.

1.2 Approach

In this thesis the interaction between an underwater vehicle and a plane rigid wall is investigated. First, a potential based boundary integral element method code was developed to create a platform from which a theoretical solution can be found. Second, part of this thesis was experimental, investigating the suction force and the pitch moment on a vehicle near a wall. The experiments were performed in the MIT Marine Hydrodynamic Water Tunnel a facility at the Department of Ocean Engineering at MIT. Following is a short description of the content of each chapter.

Chapter 2 reviews the mathematical formulation of the three different potential based formulations used in boundary integral element methods. The solution to the discretized case is shown and the influence matrix generation is explained.

Chapter 3 contains the documentation of the potential flow code development including the input parameters, the panelizer, the pressure integration and the modeling of the wall.

Chapter 4 shows the validation of the code by comparing to well known solutions as well as numerical results for different geometries.

Chapter 5 describes the experiments and their results at the MIT Marine Hydrodynamics Water Tunnel which measured forces and moments on two different underwater vehicle models.

Chapter 6 contains a discussion concerning results and tools obtained in this research project. Some suggestions for future work are explained as well.

Chapter 2

Theory

2.1 Introduction

Since Hess & Smith [2] introduced the source based boundary integral element method, panel methods have been used for a variety of applications in the fields of hydrodynamics and aerodynamics. A variety of different formulations of the boundary integral element method have been applied in both fields and most can be correct in the sense that they will converge as the panel density is increased. In the following chapter the mathematical foundation of the different boundary integral element formulations are reviewed.

2.2 Mathematical formulation

A key part of the development of boundary integral methods is the use of Green's Theorem which is used in the following derivations and explained in detail by Newman [7].

Consider the three-dimensional fluid domain V with the surface S consisting of the outer surface S_∞ and the body surface S_B shown in figure (2.1). The domain the interior to the body surface is called V′. The normal vector **n** is pointing out of the fluid V. It is assumed that the fluid in V is incompressible, invicid and irrotational and that the perturbation velocity potential ϕ is a solution to the Laplace equation in V. The body surface S_B is subject to an onset flow U_∞. The interior velocity potential ϕ' is a solution to the Laplace equation in V′. A boundary value problem can now be expressed for the two fluid domains V and V′ using the following assumptions:

- The kinematic boundary condition on the body surface S_B is $\frac{d\phi}{dn} = -\vec{U}_\infty \cdot \vec{n}$, where U_∞ is the onset flow.
- The perturbation velocity potential ϕ will diminish at the outer surface S_∞.

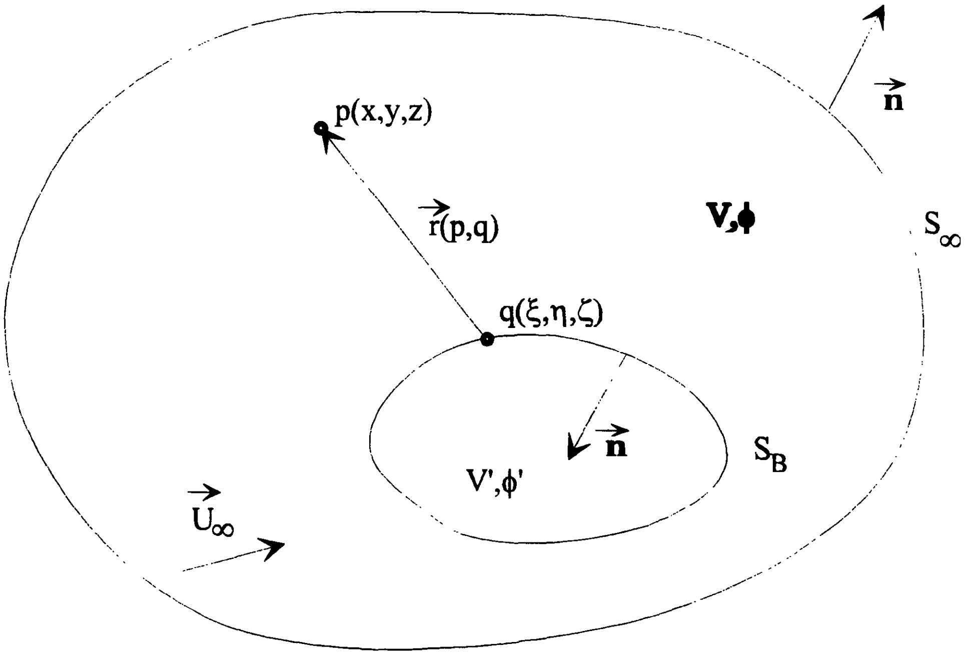


Figure 2.1: geometrical notations for application of Green's Theorem

r(q,p) is the vector from the source point q(ξ,η,ζ) to the field point p(x,y,z).

Applying Green's Theorem on the internal velocity potential ϕ' in the fluid domain V′, a boundary value problem can be stated as follows:

$$\iint_{S_B}\left(\phi'\frac{dG}{dn}-G\frac{d\phi'}{dn}\right)dS=\begin{cases}4\pi\phi'(p)\ \ (p \text{ inside } S_B)\\ 2\pi\phi'(p)\ \ (p \text{ on } S_B)\\ 0\ (p \text{ outside } S_B)\end{cases}\tag{2.1}$$

where the integral is in a principal value sense and G is a Green's function defined as:

$$G=\frac{1}{R(p,q)}$$

R(q,p) is the length of the vector $\vec{r}(q,p)$:

$$R(q,p)=|\vec{r}|=\sqrt{(x-\xi)^2+(y-\eta)^2+(z-\zeta)^2}$$

Likewise, Green's Theorem can be applied to the velocity potential ϕ in V yielding:

$$\iint_{S_B}\left(\phi\frac{dG}{dn}-G\frac{d\phi}{dn}\right)dS=\begin{cases}-4\pi\phi(p)\ (\ p \text{ outside } S_B)\\ -2\pi\phi(p)\ (\ p \text{ on } S_B)\\ 0\ (\ p \text{ inside } S_B)\end{cases}\tag{2.2}$$

Subtracting (2.1) from (2.2) results in:

$$\iint_{SB}\left((\phi-\phi')\frac{dG}{dn}-G\left(\frac{d\phi}{dn}-\frac{d\phi'}{dn}\right)\right)dS=\begin{cases}-4\pi\phi'(p)\ (\ p \text{ inside } S_B)\\ -2\pi\left(\phi(p)+\phi'(p)\right)\ (\ p \text{ on } S_B)\\ -4\pi\phi(p)\ (\ p \text{ outside } S_B)\end{cases}\tag{2.3}$$

The following looks at the case where the field point p(x,y,z) is on the body surface S_B, in which case equation (2.3) states:

$$\iint_{S_B}\left((\phi-\phi')\frac{dG}{dn}-G\left(\frac{d\phi}{dn}-\frac{d\phi'}{dn}\right)\right)dS=-\,2\pi\left(\phi(p)+\phi'(p)\right)\tag{2.4}$$

Now by setting the internal velocity potential ϕ' to various values on the surface S_B, different solutions of equations (2.4) can be derived with important physical interpretations.

Source formulation:

If the internal velocity potential is set to $\phi' \equiv \phi$ on the body S_B, equation (2.4) can be rewritten as:

$$\iint_{S_B} G\left(\frac{d\phi}{dn} - \frac{d\phi'}{dn}\right) dS = 4\pi\phi(p) \tag{2.5}$$

The term in the parentheses can be interpreted as a jump in the normal velocity across the body surface and equivalent to a source distribution with the strength σ.

$$\frac{d\phi}{dn} - \frac{d\phi'}{dn} = \sigma \tag{2.6}$$

Therefore the formulation in (2.5) is referred to as a source formulation solving for the perturbation velocity potential ϕ.

Dipole formulation

If the internal velocity potential is set to $\phi' \equiv -\phi_\infty$ on the body S_B, where ϕ_∞ is the velocity potential accociated with the onset flow $\vec{U}_\infty$ then:

$$\nabla\phi_\infty(q) = \vec{U}_\infty \quad \text{or} \quad \phi_\infty(q) = \vec{U}_\infty \cdot \vec{x}(\xi,\eta,\zeta) \tag{2.7}$$

The total velocity potential Φ, in the fluid domain V, is equal to the sum of the perturbation potential and the potential due to the onset flow:

$$\Phi = \phi + \phi_\infty \tag{2.8}$$

Using the definition of the internal velocity potential ϕ', the total potential on the surface S_B can be expressed as:

$$\Phi = \phi - \phi' \tag{2.9}$$

The boundary condition on the surface S_B approaching from the fluid domain V is:

$$\frac{\partial\phi}{\partial n} = -\vec{U}_\infty \cdot \vec{n} = -\frac{\partial\phi_\infty}{\partial n} \tag{2.10}$$

Applying the definition of internal velocity potential ϕ' in the limit approaching S_B from the internal fluid domain V′, the boundary condition can be expressed in terms of the internal velocity potential ϕ':

$$\frac{\partial \phi'}{\partial n} = -\frac{\partial \phi_{\infty}}{\partial n} \tag{2.11}$$

Applying equation (2.11) in equation (2.4) yields:

$$-2\pi\left(\phi(p) + \phi'(p)\right) = \iint_{S_B} (\phi - \phi')\frac{dG}{dn} dS \tag{2.12}$$

Applying equation (2.9) in equation (2.12) yields:

$$2\pi\left(\Phi(p) - 2\cdot\phi(p)\right) = \iint_{S_B} (\phi - \phi')\frac{dG}{dn} dS \tag{2.13}$$

Finally, by applying equation (2.8) in equation (2.13) the total potential on S_B can be expressed as:

$$\Phi(p) = 2\cdot\phi_{\infty}(q) - s\frac{1}{2\cdot\pi}\iint_{S_B} \Phi\frac{dG}{dn}\, dS \tag{2.14}$$

The definition of the total potential on S_B: $\Phi = \phi - \phi'$ can be interpreted as a jump in potential over the body surface or as a dipole distribution with strength μ. It should be noted that the freestream potential ϕ_∞ is known on S_B from equation (2.7).

The formulation in equation (2.14) is therefore referred to as a total potential or a dipole formulation.

Combined source and dipole formulation

Finally, if the internal potential ϕ' is set equal to 0 on S_B, (2.4) yields:

$$\iint_{S_B} \left(\phi(q)\frac{dG}{dn} - G\frac{d\phi}{dn}\right) dS = -2\pi\phi(p) \tag{2.15}$$

This is a source and dipole formulation solving for the perturbation velocity potential ϕ on S_B and this formulation is used in the potential flow code developed in this thesis.

2.3 Discrete Formulation

Errors associated with numerical implementation of the boundary integral are due to three types of approximations; the representation of the geometry as a set of connected panels, the discretization of the boundary integral equations and the boundary condition is only satisfied at discrete points. The errors due to these approximations decrease as the panel density increases and the solution can be shown to converge to the exact solution.

The numerical implementation of the boundary integral equation is done by discretizing the body into N quadrilateral panels each with constant strength source and dipole distribution. Using the kinematic boundary condition $\frac{d\phi}{dn} = -\vec{U}_\infty \cdot \vec{n}$ equation (2.15) yields:

$$2\pi\phi(p) - \iint_{S_B} \phi(q) \frac{1}{R^2} dS = - \iint_{S_B} \left(\vec{U}_\infty \cdot \vec{n}\right) \frac{1}{R} dS \tag{2.16}$$

In discretized form (2.16) becomes:

$$\sum_{i=1}^{N} \phi_i \cdot \left(H_{i,j} - 2 \cdot \pi \cdot \delta_{i,j}\right) = \sum_{i=1}^{N} U \cdot \vec{n}_i \cdot G_{i,j} \tag{2.17}$$

Equation (2.17) is a system of N linear independent equations with N unknowns. $H_{i,j}$ and $G_{i,j}$ are defined as:

$$H_{i,j} = \iint_S \frac{\partial}{\partial n} \frac{1}{R} dS$$

$$(2.18)$$

$$G_{i,j} = \iint_S \frac{1}{R} dS$$

$H_{i,j}$ is the induced velocity potential at the i'th panel due to a constant dipole of strength -4π on the j'th panel. $G_{i,j}$ is the induced velocity potential at the i'th panel due to a constant source strength of -4π on the j'th panel. The term $2\pi\delta_{ij}$ is the contribution from the dipole distribution on the j'th panel onto itself. The coefficients $H_{i,j}$ and $G_{i,j}$ can be evaluated analytically by geometric relationships derived by Newman [8]. The resulting expression for calculating the influence matrix rely only on the coordinates of the panel verticies. The derivation is omitted in this paper but it can be found in Newman[8]. The equations for calculating $H_{i,j}$ and $G_{i,j}$ are shown in appendix F, on page 102.

2.4 Methods of Images

A large emphasis of this project has been investigating the effect of a plane rigid boundary on an underwater vehicle operating in its vicinity. The classic approach to model a plane rigid boundary is to use the method of images, imposing a no-flux condition on the boundary using symmetry:

$$\frac{\partial \phi}{\partial \mathrm{n}} = 0$$

The method of images has been adapted to model the boundary in this project. When the induced velocity potential on the i'th panel from the dipole and source distribution on the j'th is calculated, a contribution to the velocity potential on the i'th panel from an image panel j' is included. The image panel j' is panel j mirrored in the imaginary boundary positioned at the distance h below the center line of the vehicle, see figure (2.2).

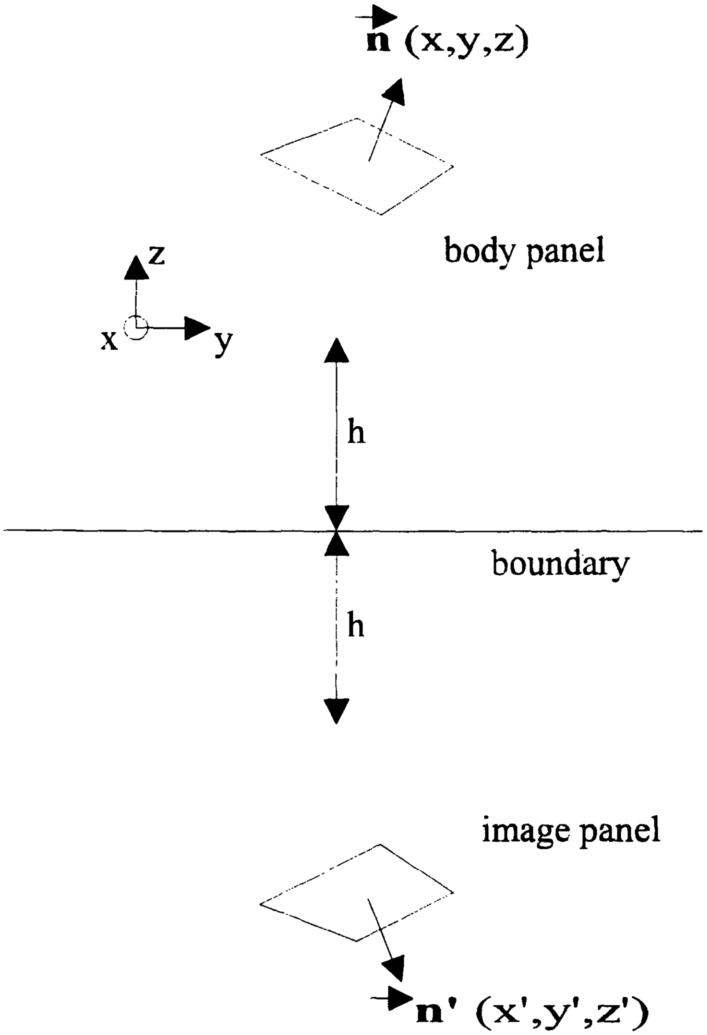


Figure 2.2: method of images notation

The dipole and source strength on panel j and on panel j' are equal in magnitude but the sign of the dipole strength on panel j′ has opposite sign of panel j. Their contribution to the flow field will, due to symmetry, cancel the perpendicular velocity component on the imaginary boundary. This cancellation is repeated for each panel pair on the body and on the image resulting in an infinite plane with the perpendicular flow component v_z equals 0.

The image panel will have the coordinates (x', y', z') where $x' = x$, $y' = y$ and $z' = -(z + 2h)$, h is the height from the vehicle centerline to the wall and the normal vector of the image panel $n'(x', y', z')$ where the components $x' = x$, $y' = y$ and $z' = -z$, (see figure 2.2).

In this chapter the mathematical foundation was reviewed for the three different potential based boundary element methods. In the next chapter the development of the BIEM code procedures is described.

Chapter 3

The Panel Code

3.1 Introduction

A modern and efficient Boundary Element Integral Method (BIEM) code was developed for this research to calculate the forces and moments on an underwater vehicle operating near the sea floor. Commercial software packages do exist like USAERO™ but they are expensive and they are complicated use due to their general applicability, so panelize even simple bodies can be a task. The developed code is a potential based BIEM code, as described in chapter 2, which uses a formulation with a constant source and dipole strength distributed over each panel and solves for the perturbation potential. This formulation was chosen because it is very robust and converges fast Lee [4] and it will furthermore be possible to model lifting surfaces by imposing a Kutta condition. The implementation of control surfaces is expected to be the next step in this ongoing project. The program was designed for ease of use and an effort was made to minimize the input parameters required by the user. The program works in a three-step process which is as follows:

1) Panelizer

   - input geometry
   - calculate corner points for quadrilateral panels coinciding with geometry
   - output to .geo-file for control of panelized geometry, the .geo-file is formatted for TECPLOT™ plotting package.

2) Setting up and solving the Influence Matrix

- calculate influence matrix and setup system of linear equations
- solve system of equation to obtain the perturbation velocity potential

3) Pressure Integration

- calculate velocities and panel pressures
- calculate forces and moments by pressure integration

Each step will be discussed in detail in the following sections.

3.2 Panelizer

The panelizer accepts any input file as long as all numbers are separated by one or more spaces or a comma and it follows the input file format which is described below. (A sample input file along with a plot of the resulting vehicle is shown in appendix I.)

Input file format:

line 1: file header

line 2: KN, NT, NX, KODE, H, RHO, NPITCH, PITCH

KN is the number of input points for the vehicle outline

NT is the number of panels along the circumference of the vehicle

NX is the number of panels in the longitudinal direction

KODE Boolean variable; if 0, no wall; if 1, wall exists

H height from wall to centerline of vehicle if KODE $\neq 1$

RHO specific density of fluid

NPITCH Boolean variable if NPITCH = 0 then no pitch (pitch angle = 0°),
if NPITCH = 1 then pitch angle = PITCH

PITCH pitch angle if NPITCH $\neq 0$

line 3: ICODE, ECCENTRICITY(a/b)

ICODE Boolean variable if ICODE = 0 then circular cross section if

ICODE ≠ 0 then cross section elliptical with axis ratio a/b = ECCENTRICITY

line 4: VX VY VZ

Velocity components

line 5: CX,CY,CZ

pivot point around which the vehicle will pitch and the point around which the moments are calculated

line 6: NCL,NCR,X_1 - X_{kn}, R_1 - R_{kn}, ESL, ESR

NCL, NCR are control variables for the slope of the vehicle outline at the ends, (see figure 3.2).

if NCL or NCR = 0 then the slope at the Left or the Right end of the outline is 0°, respectively

if NCL or NCR = 2 then the slope at the Left or the Right end of the outline is equal to ESL and ESR, respectively

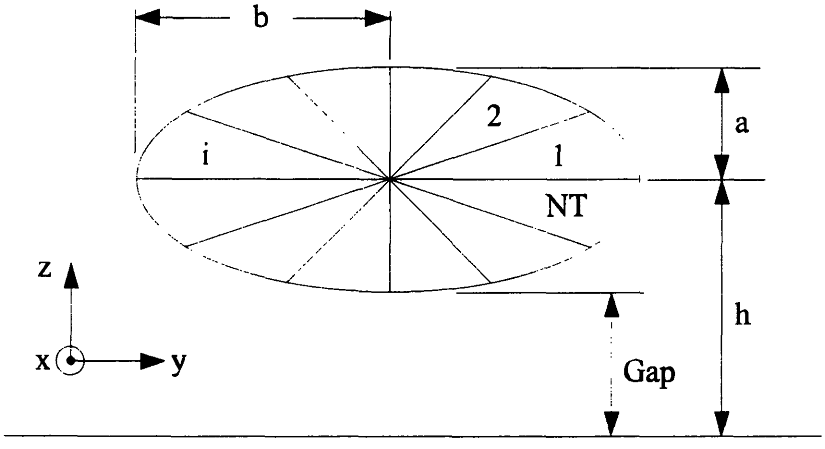
if NCL or NCR = 4 then the slope at the Left or the Right end of the curve is 90°, respectively.

Any combination of the above examples is possible

X_1 - X_{kn} : the x coordinate of the i'th station

R_1 - R_{kn} : the radius at the i'th station

ESL and ESR slope in degrees at curve end, ignored if NCL or NCR ≠ 2



Boundary

Figure 3.1: definition of major/minor axis a/b and height h

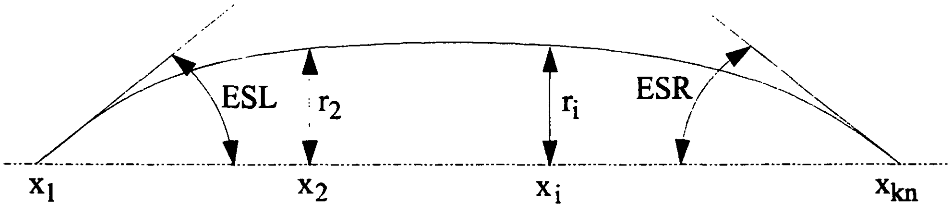


Figure 3.2: definition of input parameters

The purpose of the panelizer is to divide the geometry, provided by the user, into a finite number of quadrilateral panels which represent the original geometry as much as possible. The panelizer is designed to panelize any axisymmetric vehicle as well as vehicles with elliptical cross sections. The user defines the outline of a vehicle by supplying the radius R_i at a number of stations X_i along the length of the vehicle. A cubic-spline is then fitted to these input offsets and the radius can be evaluated at any point along the length of the vehicle. In the case of an elliptical cross section, the supplied radii are equal to the vertical axis a in figure 3.1 and the horizontal axis b is calculated from the known ratio a/b. The vehicle is divided into NT panels along the circumference which is easily done with an axisymmetrical vehicle where the azimuthical angle θ is equal to $2\pi/NT$, (see figure 3.1). With elliptical cross sections the azimuthical angles must be varied to insure equal size panels around the circumference of the vehicle. The variable azimuthical angles are found by iteration performed in the subroutine ellipse, (see appendix E). When the azimuthical angles θ_i are known, calculating the coordinates of the four vertices for each panel is a straight forward task. The panel coordinates are written to an output file in TECPLOT™ format enabling the user graphically to verify the geometry of the input vehicle. The source code for the panelizer the subroutine VEHICLE.F is shown in appendix E.

3.3 Setting up the Influence Matrix and solving the Equations

As explained in section 2.3, the discrete solution to boundary integral problem for a combined source and dipole distribution is:

$$\sum_{i=1}^{N} \phi_i \cdot \left(H_{i,j} - 2 \cdot \pi \cdot \delta_{i,j} \right) = \sum_{i=1}^{N} U \cdot \vec{n}_i \cdot G_{i,j} \qquad (3.1)$$

or in matrix notation:

$$\left[H_{i,j} \right] \left\{ \phi_j \right\} = \left\{ \vec{U} \cdot \vec{n}_j \cdot G_i \right\} \qquad (3.2)$$

The system of linear independent equations in (3.2) can be solved by using a Gaussian matrix elimination and back substitution method. This is a very laborious process considering that the matrix usual contains 10^6 elements (N = 1000), and the operational cost for solving systems of linear equations is $O(N^3)$. The elements of influence matrices for boundary integral methods are all non-zero so in solving these, one can not take the same advantage of direct solving schemes as it is possible for the sparse banded influence matrix in finite element methods. Indirect methods (iterative methods) can be used with advantage on well conditioned matrix systems like the above system with an operational cost of $O(N^2)$. However, the computational load using indirect methods can not be determined before the calculation because the computational load depends on the desired precision and on the initial guess. The matrix solver used to solve equation (3.2) in this

Research Inc., modified by Dr. H.S. Olmez of Bilkent University (Ankara, Turkey) to accommodate all I/O operations in memory. The applied matrix solver is based on the classical Gauss-Seidel elimination using a convergence acceleration technique introduced by Clark [6].

The gain in solution time using the indirect matrix solver is significant. A comparison of the solution times for a direct Gaussian matrix solver and the iterative method is shown in table 3.1 from Olmez [4]:

	Influence matrix setup	Solving matrix system	Total
Gaussian Elimination	50.0 sec	33.0 sec	83.0 sec
Iterative Solution	50.0 sec	2.5 sec	52.5 sec

Table 3.1: comparison of solution times with Gauss elimination and iterative method

A disadvantage with the indirect solver routine is that in its present form it leaves no flexibility for varying the matrix size. When ever the number of panels N changes it is necessary to manually change the array sizes in the source code and recompile the software. The problem is not that pressing because in the usual analysis sequence the same geometry will be used again and again and only the position of the geometry is varied and N is kept unchanged.

3.4 Calculation of velocity and pressure

Assuming steady and irrotational flow, the pressure p on each panel can be found using Bernoulli's equation:

$$\mathrm{p} = \tfrac{1}{2} \cdot \rho \cdot \vec{V}^2 = \tfrac{1}{2} \cdot \rho \cdot \left(\mathrm{V}_X^2 + \mathrm{V}_Y^2 + \mathrm{V}_Z^2 \right) \qquad (3.3)$$

$$p = {}^{1}\!/_{2} \cdot \rho \cdot \vec{V}^2 = {}^{1}\!/_{2} \cdot \rho \cdot \left(V_X^2 + V_Y^2 + V_Z^2\right) \tag{3.3}$$

Using the local coordinate system Bernoulli's equation (3.3) can be modified yielding equation (3.4):

$$p = {}^{1}\!/_{2} \cdot \rho \cdot \left(\left(\frac{\partial\phi}{\partial x`}\right)^2 + \left(\frac{\partial\phi}{\partial y`}\right)^2 \right) \tag{3.4}$$

where x′ and y′ are in local tangent plane.

Analytic solutions to the velocity terms $\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}$ and $\frac{\partial\phi}{\partial z}$ can be found, but because of the singularities on the body, I found it preferable to evaluate the velocities numerically. The numerical differentiation is done by fitting a quadratic equation to the velocity potential on the i'th panel and the two adjacent panels.

$$\phi_i = A\xi_i^2 + B\xi_i + C \quad , i = 1,3 \tag{3.5}$$

When combining the three equations in (3.5), expressions for the constants A and B are found as (3.6):

$$A = \frac{(\phi_1 - \phi_2)\cdot(\xi_2 - \xi_3) - (\phi_2 - \phi_3)\cdot(\xi_1 - \xi_2)}{(\phi_1^2 - \phi_2^2)\cdot(\xi_2 - \xi_3) - (\xi_2^2 - \xi_3^2)\cdot(\xi_2 - \xi_1)}$$

$$(3.6)$$

$$B = \frac{(\phi_1 - \phi_2)\cdot(\xi_2 - \xi_3) - A\cdot(\xi_1^2 - \xi_2^2)\cdot(\xi_2 - \xi_3)}{(\xi_1 - \xi_2)\cdot(\xi_2 - \xi_3)}$$

By differentiating (3.5) with respect to the spatial variable ξ, the velocity component u is expressed as:

$$u = \frac{\partial\phi}{\partial\xi} = 2\cdot A\cdot\xi + B \tag{3.7}$$

The calculations are eased due to the fact that only the magnitude not the direction of the velocity on each panel is needed to calculate the pressure on the panel. Two components for the velocity are calculated at each panel, one in the longitudinal direction and one in the tangential direction respectively, (see figure 3.4). The two vectors obtained are not orthogonal and it is necessary to derive two orthogonal velocity components, which is done using equation (3.8).

$$\begin{aligned} \boldsymbol{u} &= \left|\vec{\boldsymbol{b}}\right| \cdot \cos\theta \\ &\cdot \\ \boldsymbol{v} &= \left|\vec{\boldsymbol{a}}\right| - \left|\vec{\boldsymbol{b}}\right| \cdot \sin\theta \end{aligned} \qquad (3.8)$$

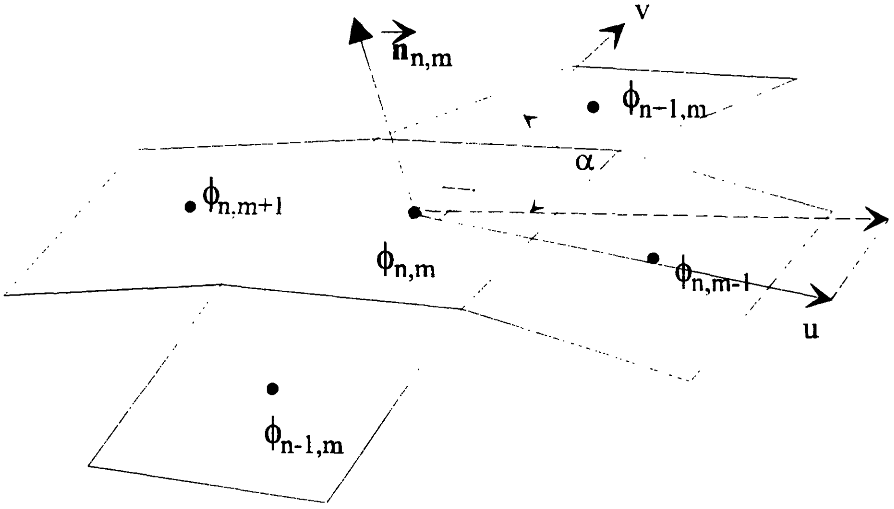


Figure 3.4: Local velocity components

Having two orthogonal components for the velocity on each panel, the local pressure on a panel can be calculated by using equation (3.5). The subroutine calculating the velocity components VELO.F is listed in appendix D.

3.5 Pressure Integration

The six components of the force vector on a body with surface S are defined by the integral (3.9) and (3.10) where $\vec{\boldsymbol{n}}$ is the normal vector pointing out of the body and p the local pressure:

$$\vec{F} = -\iint_S p\vec{n}\,dS \tag{3.9}$$

$$\vec{M} = -\iint_S p(\vec{r} \times \vec{n})\,dS \tag{3.10}$$

In the discretized case equation (3.9) and (3.10) can be expressed as:

$$\vec{F} = -\sum_{i=1}^{N} p_i \vec{n}_i dA_i \tag{3.11}$$

$$\vec{M} = -\sum_{i=1}^{N} p_i \left(\vec{r}_i \times \vec{n}_i\right) dA_i \tag{3.12}$$

where A_i is the area, $\vec{n}_i$ is the normal vector and $\vec{r}_i$ is the spatial vector from the point O around which the moments are calculated to the centroid of the i'th panel respectively. The forces and moments are calculated in the subroutine PRESS.F, (see appendix C). For convenience the force and moment coefficients are calculated as well. They are defined as:

Force coefficient $$C_f = \frac{\text{Force}}{0.5 \cdot \rho \cdot U^2 \cdot S} \tag{3.13}$$

Moment coefficient $$C_m = \frac{\text{Moment}}{0.5 \cdot \rho \cdot U^2 \cdot S \cdot l} \tag{3.14}$$

This chapter described the procedures in the developed BIEM code. In the next chapter numerical results from the code will be compared to well known potential flow cases. Numerical results on other geometries produced by the developed code will be presented as well.

Chapter 4

Numerical Results

4.1 Testing the Code

Different tests were done to compare the numerical results with known analytical results to verify the validity and accuracy of the panel code. Comparisons were also done between slender body theory and the panel code using different geometries

4.2 Tangential Velocity on a sphere

The initial test of the accuracy of the BIEM code was performed with the well known result of the tangential velocity on a sphere in uniform flow. The onset flow is in the positive x direction. The sphere is panelized with one pole facing the flow. This is also the case when axisymmetric vehicles are panelized. (see figure 4.3 through figure 4.5). Figure 4.1 shows the calculated results for different panel densities and the theoretical results. The angle theta is zero at the pole and 90 degrees at the equator. The results are symmetric for the aft half of the sphere. The calculated results show good correlation with the theory especially when more that 400 panels are used, which is a good indication that the code works well.

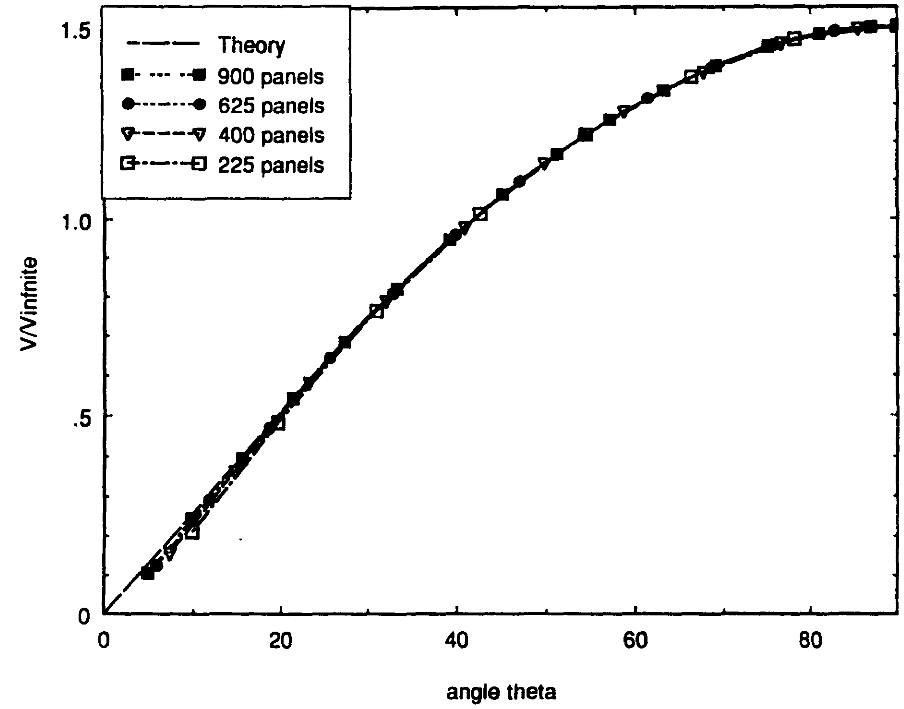


Figure 4.1: tangential velocity vs. angle θ on sphere

4.3 Suction force acting on sphere near wall

In this section the case of a sphere near a wall is investigated. For this particular case we have an approximate solution given by Milne-Thomson [5]. Analyzing a sphere near a wall is similar to analyzing a vehicle near a wall except for the differences in the geometry and the fact that there is no analytic expression for an arbitrary vehicle. Figure 4.2 shows the force coefficients vs. gap/diameter. When the gap is greater than 10-15% of the sphere's diameter, the calculated force coefficients and the theoretical approximation agree very well. When the gap decreases the calculated results converge towards a value approximately 80% higher than the theoretical approximation. The theoretical approximation of the velocity potential ignores a term of $r^6/4h^6$ where r is the radius of the sphere and h is the distance from the center of the sphere to the wall. This approximation is accurate while the gap is of the same order as the radius but when the gap gets smaller the error grows significantly. The suction force coefficient is defined as follows:

$$C_S = \frac{\text{Suction force}}{0.5 \cdot \rho \cdot V^2 \cdot D^2}$$

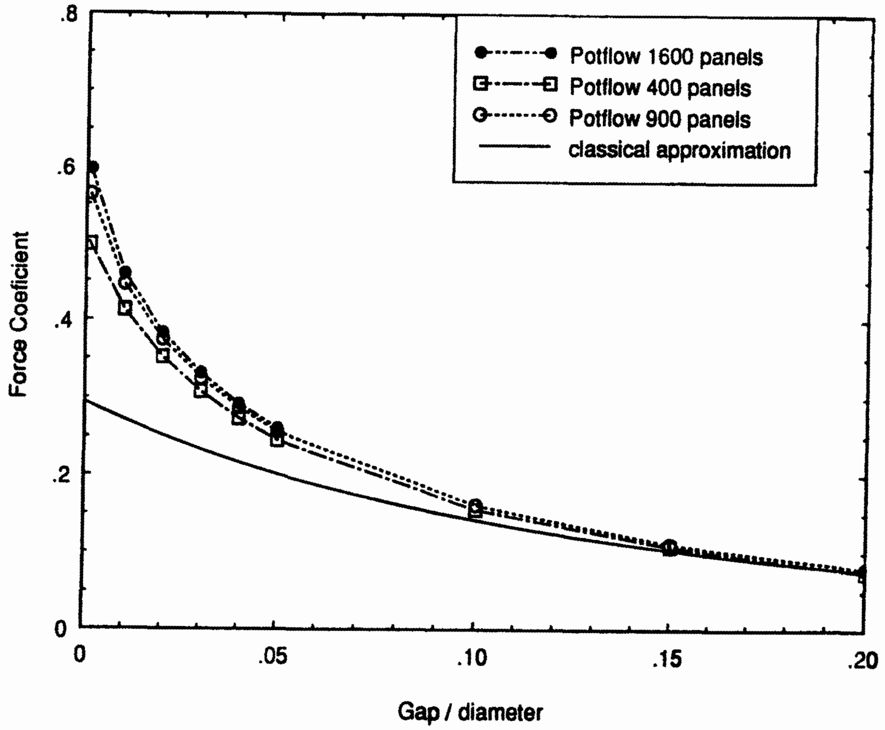


Figure 4.2: suction force coefficient vs. gap/diameter for sphere

Figure 4.3 through 4.5 show the discretizations used for the calculations.

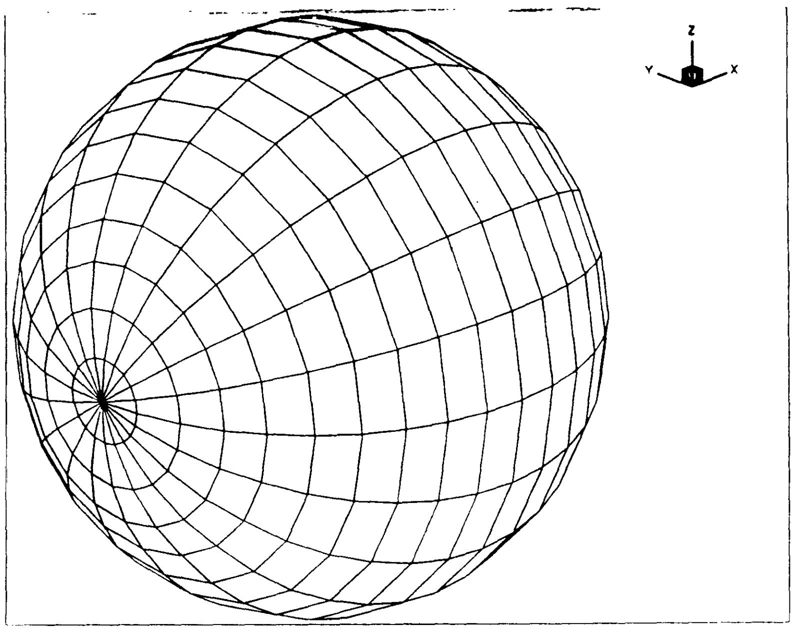


Figure 4.3: 400 panel discretizations of sphere

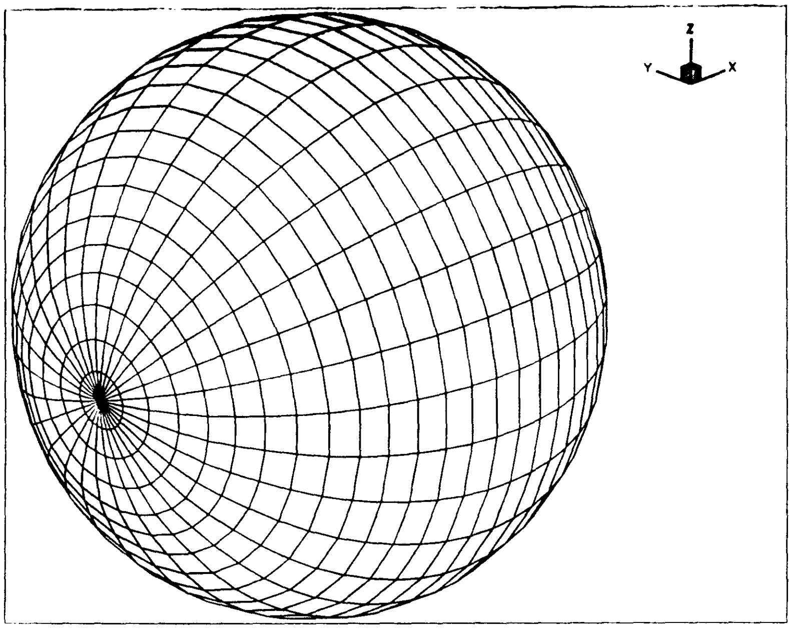


Figure 4.4: 900 panel discretizations of sphere

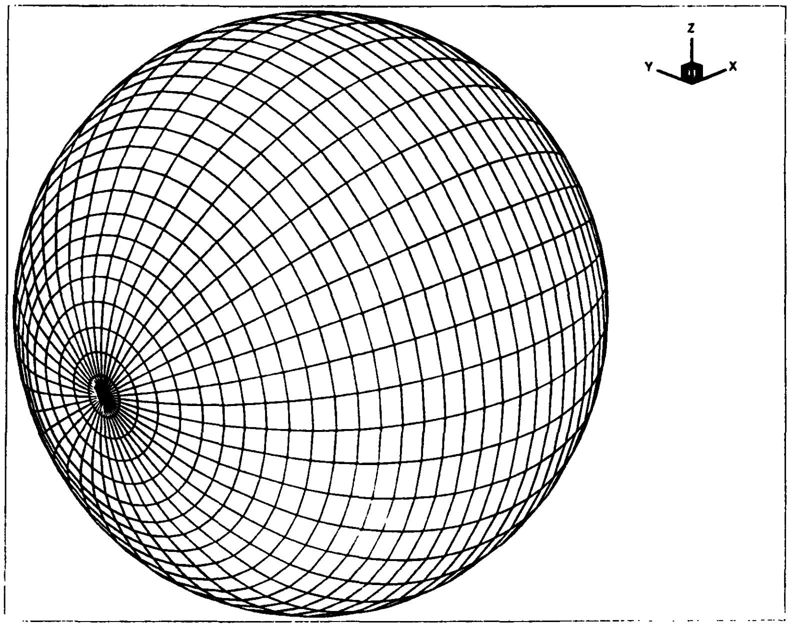


Figure 4.5: 1600 panel discretizations of sphere

4.4 Comparison with slender body theory

This test involves a spheroid which shape is similar to that of many underwater vehicles. We will compare panel method results with predictions from the slender body theory explained by Newman [7]. A spheroid with a length to diameter ratio of 40 is used as the first geometry. It is very slender, figure 4.6 shows the suction force coefficient vs. gap / vehicle diameter. The slender body results and the potential flow code results agrees quite well as expected.

Figure 4.7 shows the suction force on a less slender spheroid with a length to diameter ratio of 8, which is a normal length to diameter ratio for underwater vehicles. In this case the results from the slender body theory are quite different from the results calculated from the potential flow code. In this case the slender body result differs due the fact that the body is not slender enough. The slenderness of the vehicle is not the only factor determining the correctness of the slender body theory, the shape is also very important as this next case illustrates, where a cylindrical vehicle is considered. The cylinder's length to diameter ratio is 40 and both ends of the cylinder are hemispherical. Figure 4.8 shows the suction force coefficient vs. gap/diameter for this vehicle near a wall. In this case the slender body theory agrees well with the potential flow code when the gap is large, but disagrees when the gap gets smaller than the diameter of the vehicle. This disagreement is due to the fact that the slender body theory does not account for the effects of three dimensional flow near the ends. The three different discretizied bodies are shown in figure 4.9 through figure 4.11.

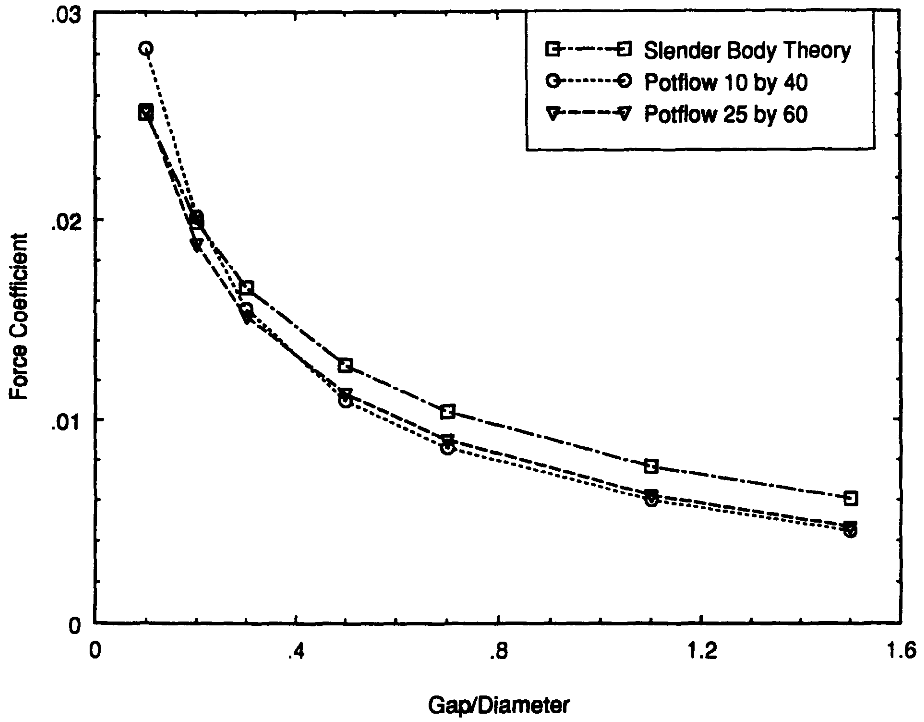


Figure 4.6: suction force coefficient on spheroid length to diameter ratio of 40

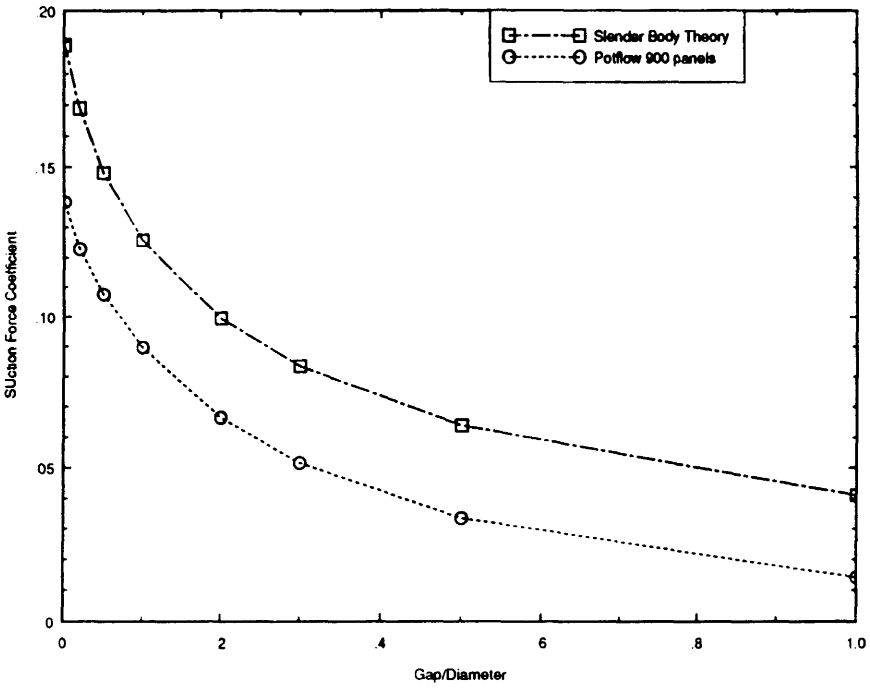


Figure 4.7: suction force on spheroid with length to diameter ratio of 8.0

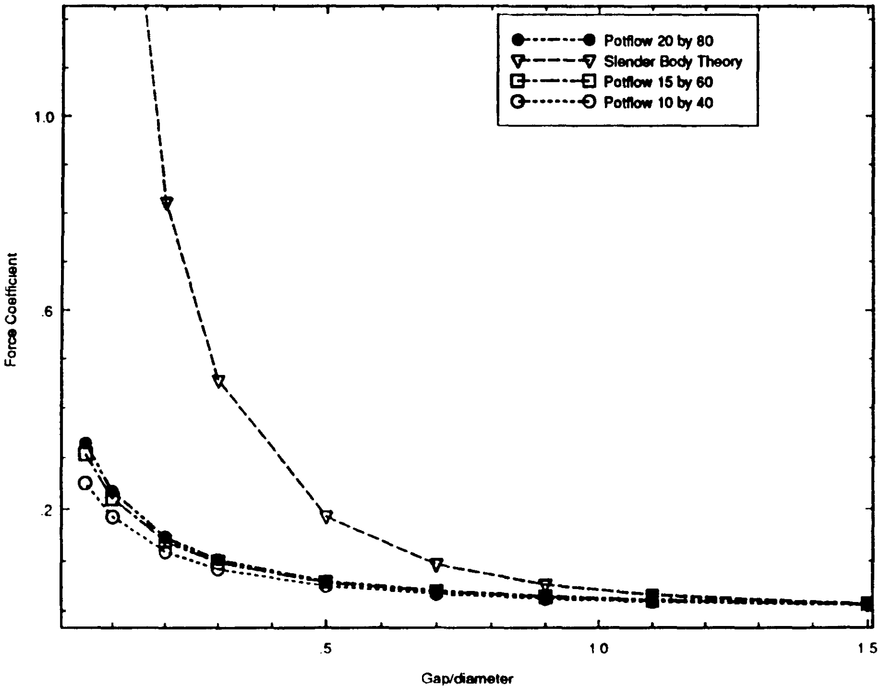


Figure 4.8: suction force on cylinder with length to diameter ratio of 40

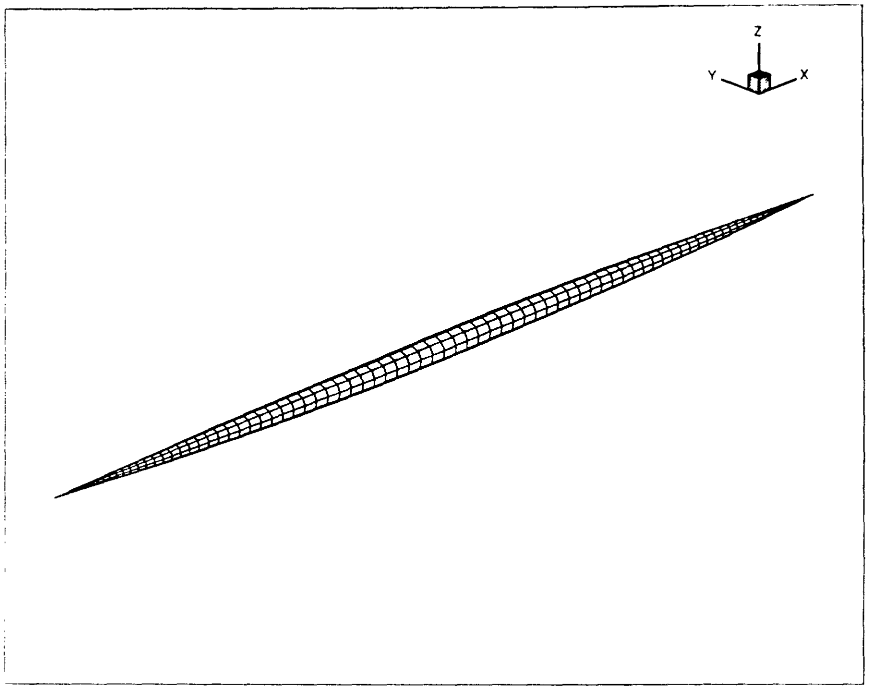


Figure 4.9: 900 panel discretizations of spheroid with length to diameter ratio of 40

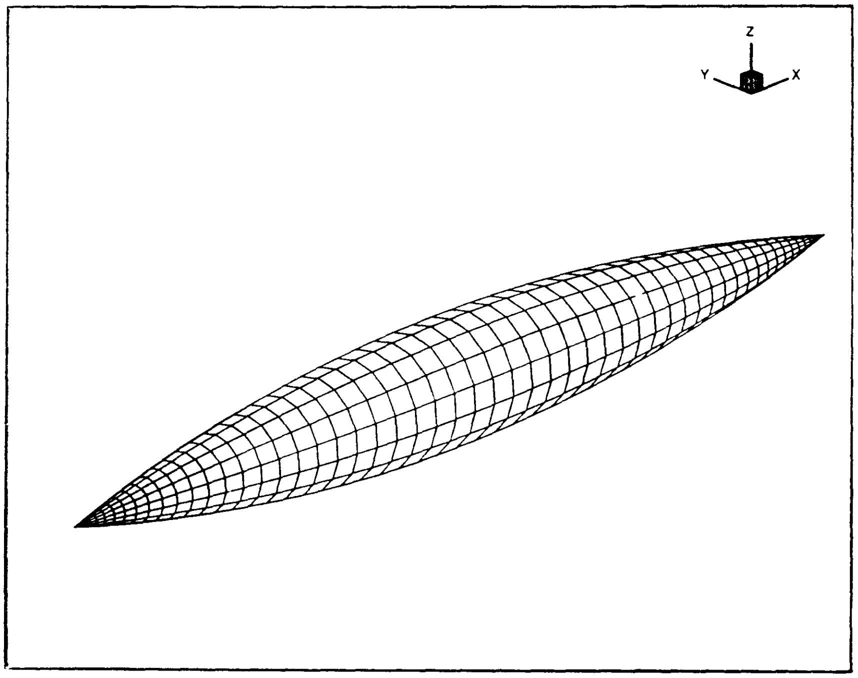


Figure 4.10: 900 panel discretizations of spheroid with length to diameter ratio of 8.0

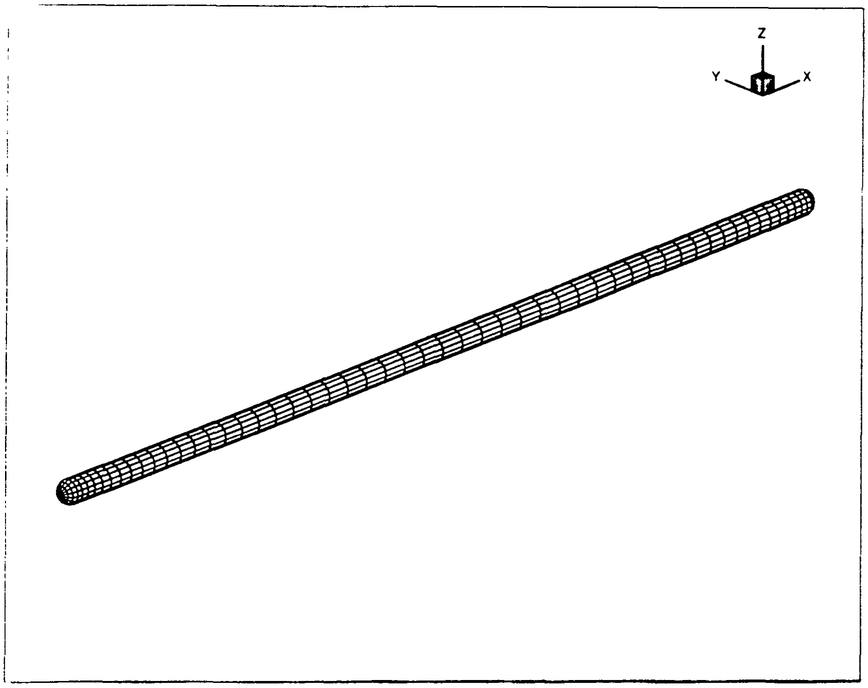


Figure 4.11: 900 panel discretizations of cylinder with length to diameter ratio of 40

4.5 Nonsymmetrical vehicles

The previous cases have all involved fore-and-aft symmetrical vehicles. In the following section, two similar asymmetrical vehicles are numerically tested: the S. C. Draper Laboratory (CSDL) vehicle model 666 and a Woods Hole Oceanographic Institute (WHOI) underwater vehicle. Models of both vehicles were also tested in the MIT Marine Hydrodynamic Water Tunnel, the experiments are described in chapter 5. Both vehicles are AUV's under development at the two institutions. Figure 4.12 shows the suction force coefficient vs. gap/diameter for the two vehicles and a spheroid is shown as a reference. All three vehicles have a length to diameter ratio of 6. The suction force coefficients for the two vehicles are quite similar, while the suction force for the spheroid is significantly lower. The lower suction force on the spheroid is due to the fact that the spheroid has less volume compared to its diameter than the other two vehicles and resulting in a smaller part of the body area near the small gap with the low pressure.

The pitch moment coefficient , defined in equation (3.14)), associated with the suction force is shown in figure 4.13. A negative pitch moment means that the bow is forced towards the wall. The results for the spheroid is trivial due to fore-and-aft symmetry. The results for the two vehicles, however, are quite different. The pitch moment coefficient on the WHOI vehicle is more that twice that of the CSDL vehicle. The differences are due to the differences in the fore-and-aft geometry. The WHOI vehicle has a very blunt forward body, and this moves the center of effort further forward on the WHOI vehicle than on the CSDL vehicle. A rough estimate indicates that the center of effort moves forward about 1/15th of the vehicle length on the CSDL vehicle and twice that on the WHOI vehicle. The two discretizied vehicles are shown in figure 4.14 and 4.15.

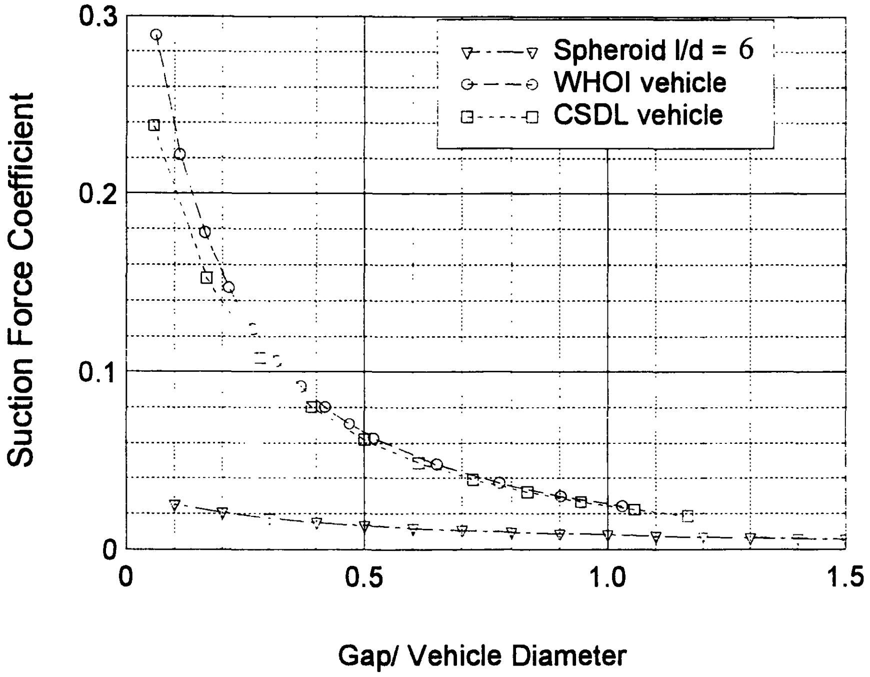


Figure 4.12: suction force coefficients on vehicles near wall

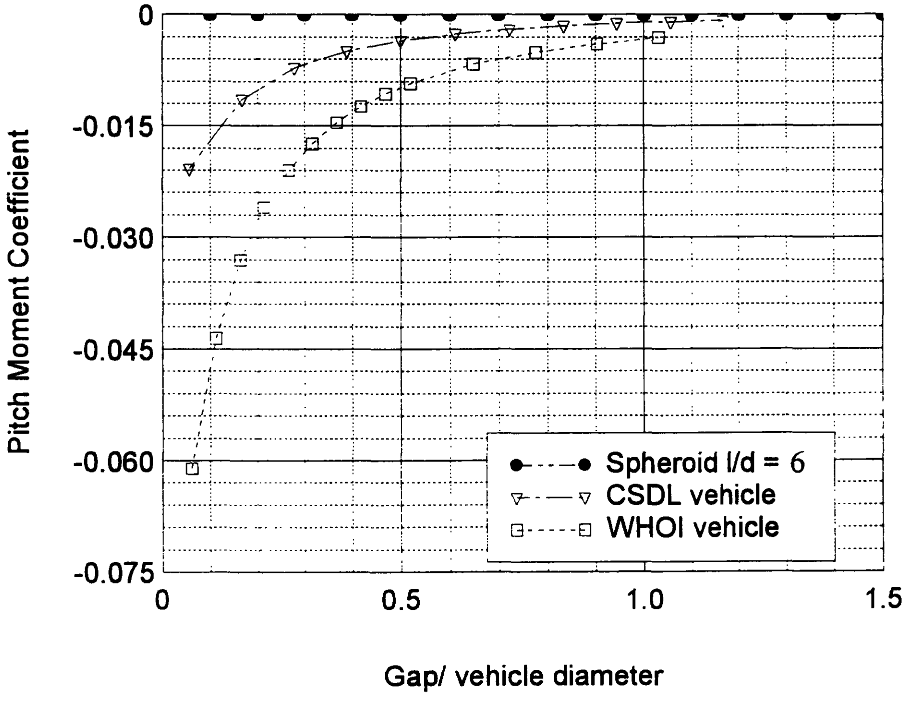


Figure 4.13: pitch moment coefficient on vehicles near wall

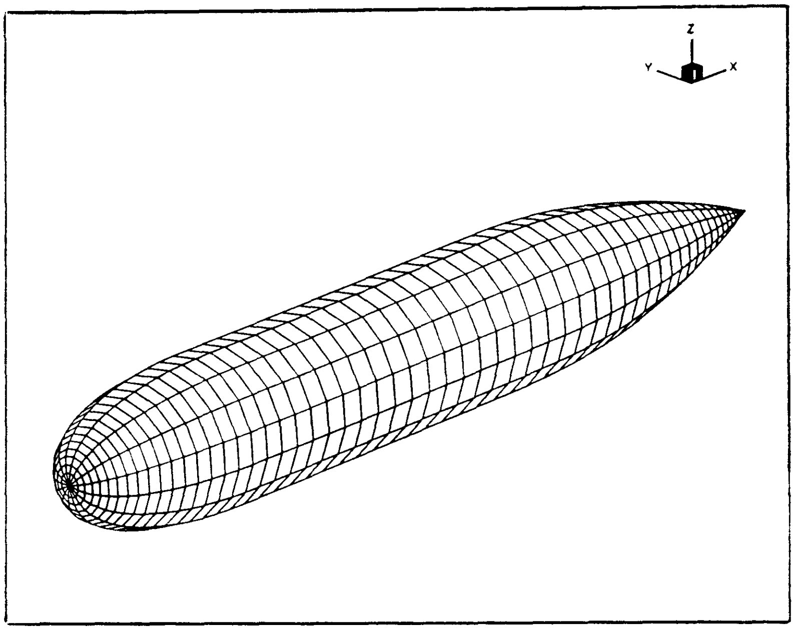


Figure 4.14: 900 panel discretization of the CSDL vehicle

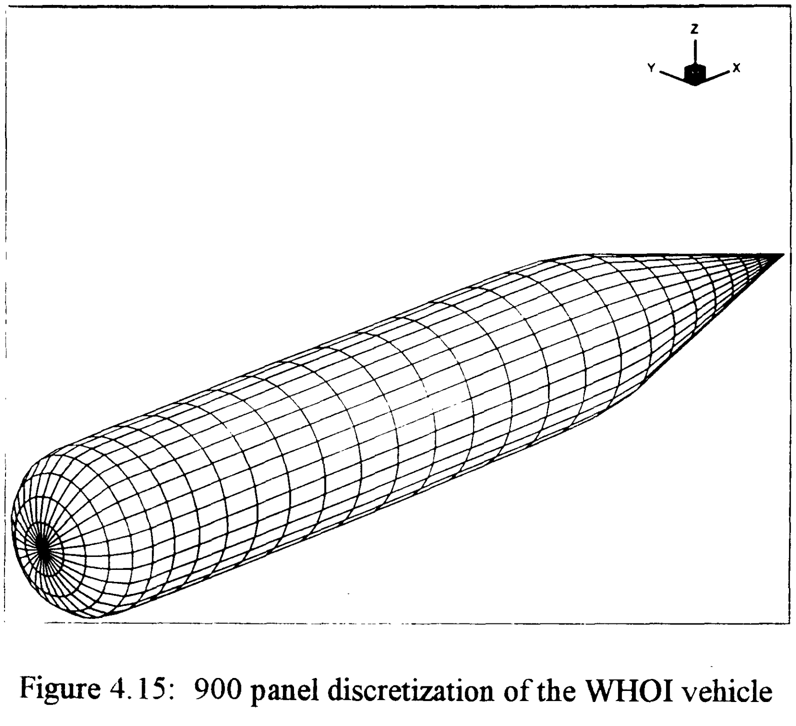


Figure 4.15: 900 panel discretization of the WHOI vehicle

In this chapter the numerical results from the BIEM code was compared to well known potential flow cases. Numerical results for the two AUV models were presented as well. In the next chapter the two MIT Marine Hydrodynamic Water Tunnel experiments of the CSDL and WHOI models are described.

Chapter 5
Experiments

5.1 Introduction

Two experiments were conducted in the MIT Hydrodynamics Water Tunnel (MITHWT) using models of the CSDL and WHOI vehicles. In these experiments the forces and moments were measured on a model with different distances to the wall and in the case of the CSDL vehicle they where also measured at different pitch angles. The results from the CSDL model experiment are being used in an ongoing project at Charles Stark Draper Laboratory Inc. (CSDL) where they are developing algorithms for the control and navigational systems used in Autonomous Underwater Vehicles (AUV's). The results from the WHOI model experiment are being used in the development of the control components of the vehicle. These tests were conducted to assess the real fluid effect on the flow around the vehicles and thereby get a measure of the applicability of the numerical results in real life situations. In the following sections the test models and the test equipment are described along with the results from the tests.

5.2 The WHOI test model

The test model for the WHOI experiment was a 1:10 scale of a AUV currently being developed at the Woods Hole Oceanographic Institution. The WHOI model is 2 inches in diameter and 12 inches long. Its profile is shown in figure 5.2. This model was also used for other experiments outside the scope of this thesis and these required the small size.

5.3 The CSDL test model

The test vehicle model used for the CSDL experiment was a modified configuration of the 666 model supplied from CSDL (UUV1). The full scale vehicle is 27 feet long and has a length to diameter ratio of 7.365. During the previous tests with the WHOI model suction forces were in the order 1 pound. Scaling the WHOI model to double size meant the expected forces were around 8 pounds. The larger force provides a better signal to noise ratio. The CSDL model diameter was therefore chosen to be 4.0 inches for the above reasons. In order to simulate high pitch angles in the MITHWT without risking significant influence from the far wall of the 20 by 20 inches test section, it was necessary to shorten the vehicle. To retain similarity of the end shapes with the original vehicle, only the cylindrical middle part of the vehicle was shortened while the nose and tail sections were kept unchanged. The CSDL model was chosen to be 24.0 inches long, thus the length to diameter ratio is 6.0. The changes in appearance due to these modifications are displayed in figure 5.2 which shows the profiles of the CSDL test model, the scaled down and shortened UUV1 and the WHOI test model.

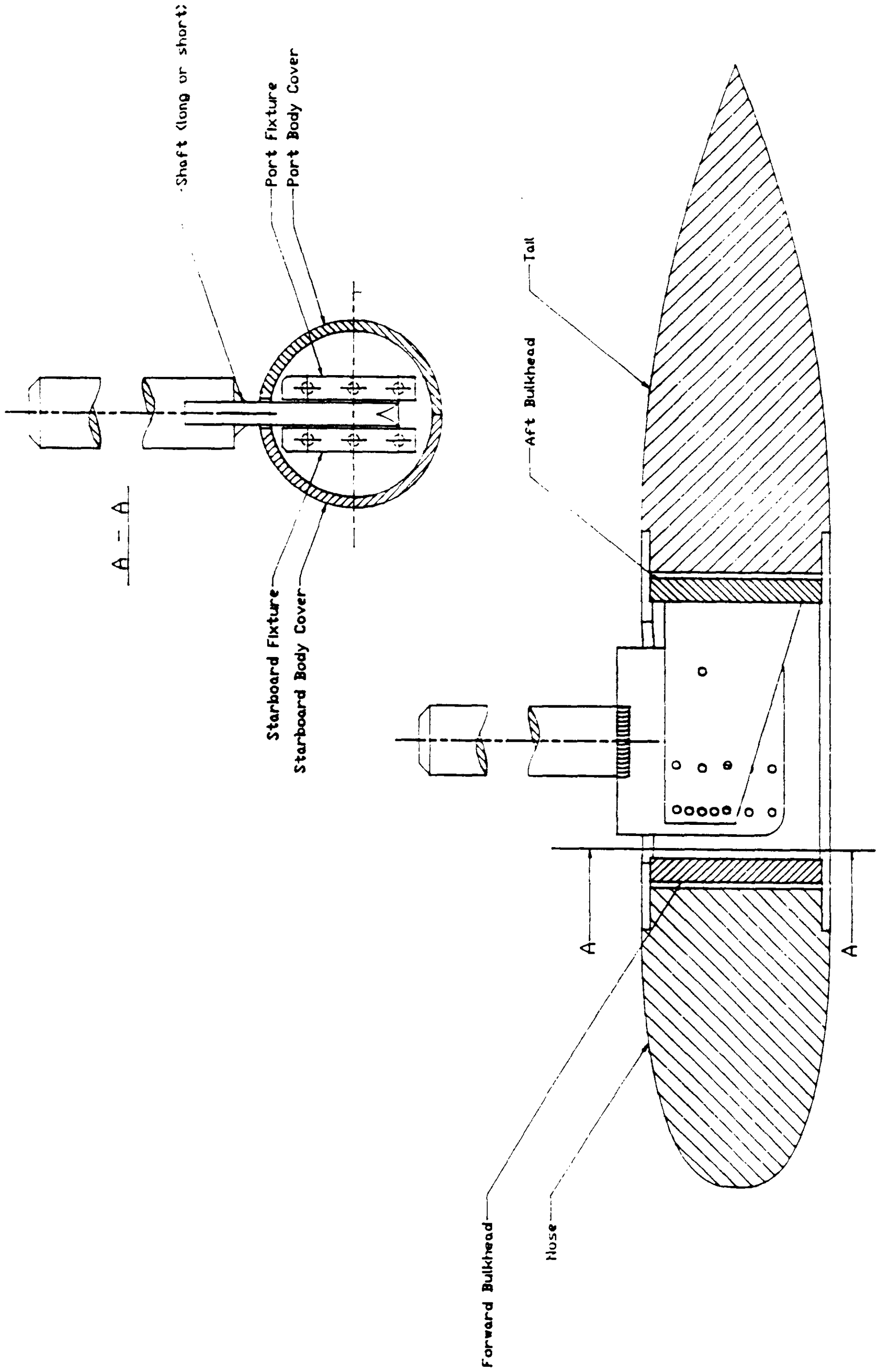


Figure 5.1: CSDL test model components

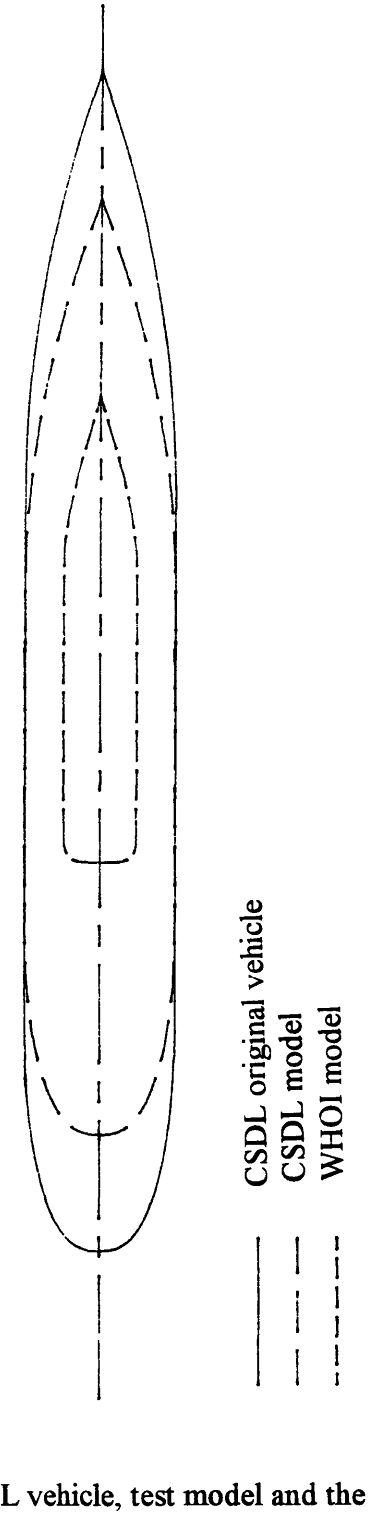


Figure 5.2: outline of original CSDL vehicle, test model and the WHOI test vehicle

5.4 The CSDL test model and its construction

The CSDL model consists of three main parts; nose, tail and main body. A plot of the main parts of the CSDL model is shown in figure 5.1. Both the tail and nose sections are attached to the main body by six machine screws, a feature that eases the effort and shortens the time if different nose or tail sections are to be tested. The main body contains the mechanism by which the model is positioned in the preferred height and pitch angle. The model can be positioned at four heights on the mounting shaft. At each height the model can be positioned in seven different pitch angles: ±15°, ±10°, ±5° and 0°. The flat part of the mounting shaft slides between the port fixture and the starboard fixture and is then fixed at the desired height by positioning a bolt in the appropriate pivot. The model is then positioned at the desired pitch angle, the holes are lined up and the fastening bolt is positioned. Two shafts were made with different lengths to enable a wider range of testing heights. The range for each shaft is only 1.5 inches so the total range is 3.5 inches. This changes the height from the model centerline to the wall from 2.56 to 6.06 inches. The positioning system is very simple and rugged with a high safety margin. The most obvious draw back is that all model position changes have to be performed from inside the water tunnel. This adds to the testing time but the possibilities of downtime due to system failure are highly reduced and it is easier to plan for extra time for changing the setup than for repairs. With an experienced operator a change of pitch and height can be done in less than half an hour.

5.5 The MIT Hydrodynamics Water Tunnel

The MIT Hydrodynamics Water Tunnel, designed by Professor Frank Lewis, was built in 1939. At that time it had a 20 inch diameter open jet test section. The water tunnel was originally designed for propeller testing as its original name, Propeller Testing Tunnel, indicates. The water tunnel was redesigned in 1968 to accommodate a wider variety of experiments. The rebuilding resulted in a 20 by 20 inch square test section with easily removable windows on all four sides of the test section. A storage tank with a high-capacity pump was installed for fast drainage and filling, making it possible to reuse the water and treat the water with rust inhibitors. The velocity in the test section can be varied from approximately 2 ft/sec to 34 ft/sec. A schematic view of the Hydrodynamics Water Tunnel in its present form is shown in figure 5.3.

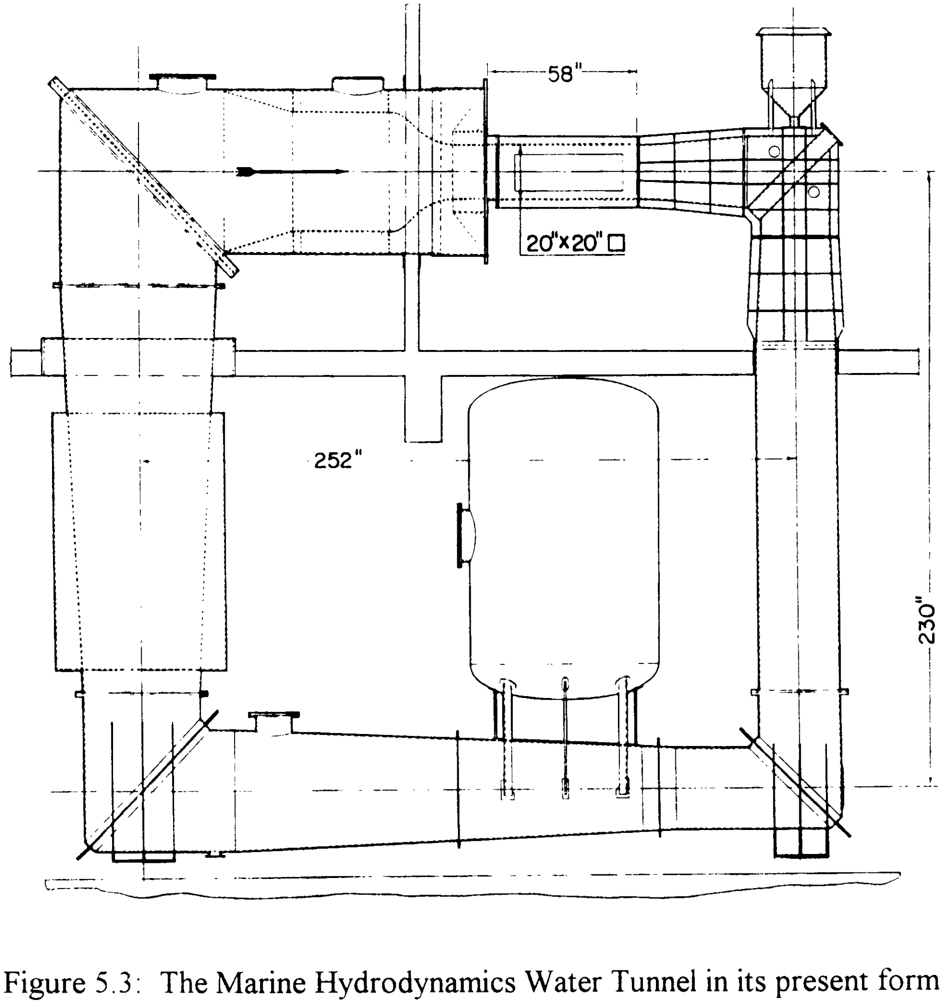


Figure 5.3: The Marine Hydrodynamics Water Tunnel in its present form

5.6 The dynamometer

A six axis dynamometer was used to measure the forces and moments on the model. The dynamometer is permanently mounted on a window that fits any of the four openings in the tunnel test section. The dynamometer and its main components are shown in figure 5.4. The dynamometer frame serves as the base of six load cells. The load cells are connected to the frame by flexure rods and again through flexure rods to the dynamometer sleeve. The model is fastened through the dynamometer window with a 1-1/2 inch shaft into the sleeve. The flexure rods have a short narrow part to minimize transverse forces carried by the load cells. The load cells measure only axial loads, thus any transverse forces will be accounted for only in the calibration procedure. The model is "floating", connected to the surroundings only through flexure rods and the load cells which measures all forces and moments on the model. To insure water tightness around the shaft going through the dynamometer window an oil/water seal is mounted in a rubber bellows attached to the window. Table 5.1 shows the nominal load capacity and position of the used load cells.

Nominal Load	Position
50 lb.	1
50 lb.	2
100 lb.	3
25 lb.	4
25 lb.	5
25 lb.	6

Table 5.1: dynamometer configuration

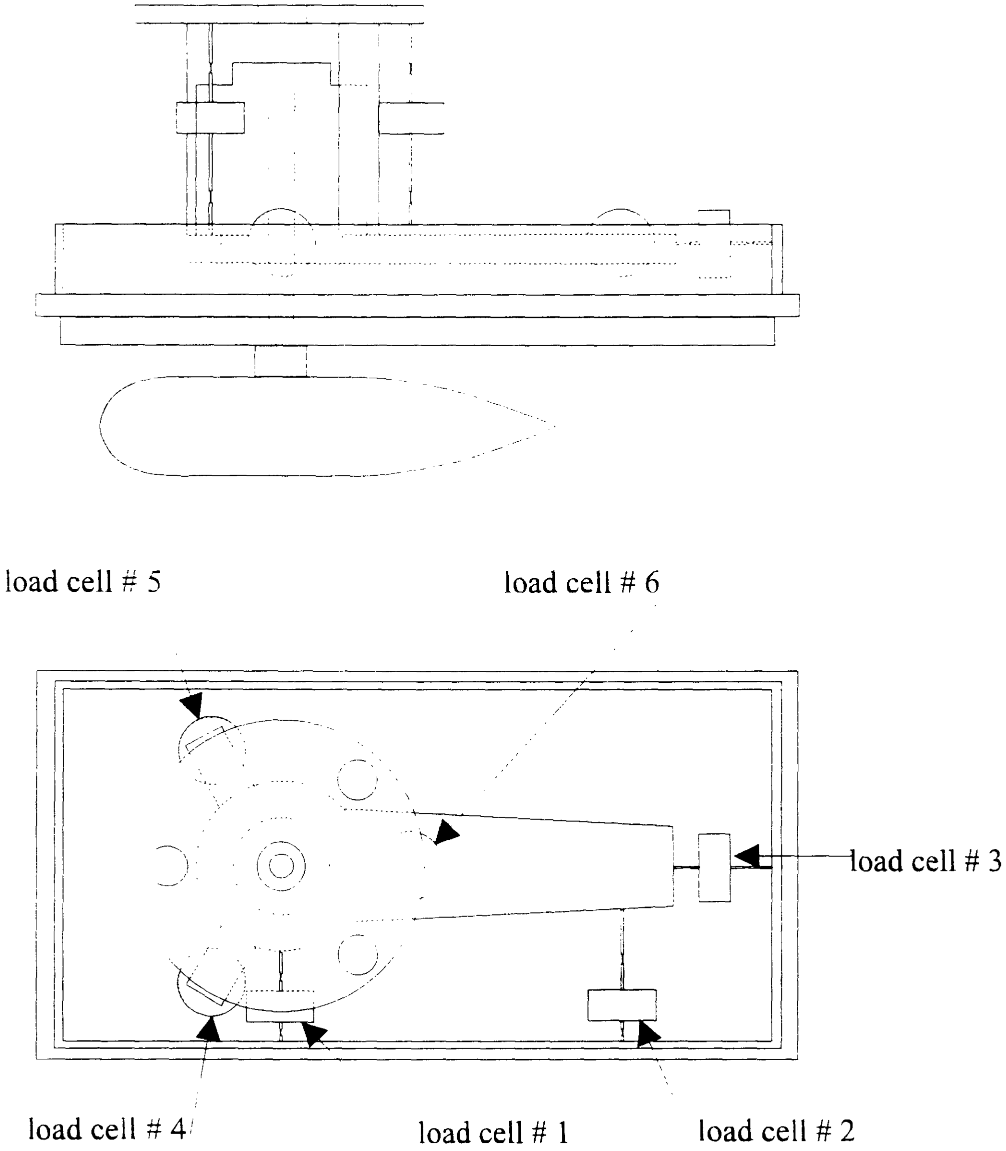


Figure 5.4: dynamometer and its main components

5.7 Calibration

In this test there is little interest knowing the loads on each load cell. What is of interest. however, are the total forces and moments applied on the model. It is therefore necessary to calibrate the dynamometer so that the reading from each of the six load cells can be combined and the total forces and moments derived.

The dynamometer has six degrees of freedom and any arbitrary load on the model will give a reading on each of the six load cells. the six components in the load vector $\vec{l}$. To determine one of the six elements f_i of the force vector $\vec{f}$ it is necessary to take the contribution from each of the load cells into account. As an example, a force in the positive z- direction is considered. Ideally there would be only the three contributions from the load cells in position 4. 5 and 6. Due to misalignments and cross talk (the fact that the load cells will register some off axial load) all six load cells contribute. The contributions are represented by the elements c_{31} through c_{36} in the calibration matrix **C**.

$$f_3 = \sum_{i=1}^{6} c_{3i} \cdot l_i \qquad (5.1)$$

This can now be repeated for the other five elements in the force vector $\vec{f}$:

$$f_j = \sum_{i=1}^{6} c_{j,i} \cdot l_i \qquad (5.2)$$

and thus we have a system of linear equations:

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{16} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} & c_{26} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} & c_{36} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} & c_{46} \\ c_{51} & c_{52} & c_{53} & c_{54} & c_{55} & c_{56} \\ c_{61} & c_{62} & c_{63} & c_{64} & c_{65} & c_{66} \end{bmatrix} \begin{Bmatrix} l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_6 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{Bmatrix} \qquad (5.3)$$

By applying N different load cases we obtain the following matrix system:

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{16} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} & c_{26} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} & c_{36} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} & c_{46} \\ c_{51} & c_{52} & c_{53} & c_{54} & c_{55} & c_{56} \\ c_{61} & c_{62} & c_{63} & c_{64} & c_{65} & c_{66} \end{bmatrix} \begin{Bmatrix} l_1^1 \cdots l_1^N \\ l_2^1 \cdots l_2^N \\ l_3^1 \cdots l_3^N \\ l_4^1 \cdots l_4^N \\ l_5^1 \cdots l_5^N \\ l_6^1 \cdots l_6^N \end{Bmatrix} = \begin{Bmatrix} f_1^1 \cdots f_1^N \\ f_2^1 \cdots f_2^N \\ f_3^1 \cdots f_3^N \\ f_4^1 \cdots f_4^N \\ f_5^1 \cdots f_5^N \\ f_6^1 \cdots f_6^N \end{Bmatrix} \qquad (5.4)$$

In the case of N = 6, or in other words six different load cases, assuming the six load cases are linearly independent, a solution to (5.4) can be found by simple matrix algebra:

$$\mathbf{C} = \mathbf{FL}^{-1} \qquad (5.5)$$

To minimize the possibility of errors, a larger number of load cases are applied, N > 6. The system is now an over determined system of linear equations and a solution can be found using the method of least squares. This means determining **C** while minimizing the norm:

$$\left\| \mathbf{F}^{\mathrm{T}} - \mathbf{L}^{\mathrm{T}}\mathbf{C}^{\mathrm{T}} \right\|^2 \qquad (5.6)$$

The calibration was done by applying a set of linearly independent, but known, load cases and reading the output from the load cells. The loads were applied by means of weights applied to an arm mounted in the dynamometer, The load was directed in the appropriate direction using lines and pulleys. A listing of the nine calibration load cases used in the calibration is shown in table 2. The absence of a simple way of applying a pure moment made it necessary to couple each of the three moments with at least one force.

Test Sequence #	1	2	3	4	5	6	7	8	9
Fx	x			x	x				
Fy		x				x	x		
Fz			x					x	x
Mx						x		x	
My				x					x
Mz					x		x		

Table 5.2: calibration sequences

Three different loads were applied for each load point and force direction so 27 cases were recorded. A computer program SJCAL.FOR was written to perform the least square solution using the columns of **L** and **F** as input. The source code is shown in appendix B. When the calibration matrix is known, calculation of the forces and moments is done by simple algebra, using equation (5.3).

The calibration matrix **C** was found to be:

.00003	.00021	.06792	.00017	.00006	-.00015	[lbs/counts]
.02932	.02592	-.00021	-.00007	.00019	.00019	[lbs/counts]
-.00165	-.00125	-.00089	-.02492	-.01942	-.01453	[lbs/counts]
-.23396	-.21145	.00291	-.14275	.11353	-.00272	[in-lbs/counts]
-.00309	.00190	.53766	-.08207	-.06203	.22782	[in-lbs/counts]
-.00137	.46525	-.00953	-.00143	.00098	.00212	[in-lbs/counts]

The following should be noticed about **C**:

Forces in x-direction, upstream in the tunnel. are mainly carried by load cell #3. Element c_{13} is significantly larger than the other elements in row 1. Forces in y-direction, sideways towards port on the model, are mainly carried by load cells #1 and #2. The vertical forces are mainly carried by load cell #4, #5 and #6.

5.8 Data acquisition

The load cells used on the dynamometers were all of the full bridge type thus they consist of a full Wheatstone bridge where the four resistors are strain gauges. As load is applied to the load cell the resistance of the strain gauges in the bridge changes, causing the voltage over the output terminals to change.

The load cells generate an output of 2 millivolt per volt excitation applied over the bridge at nominal load. Maximum allowed excitation voltage is 12 volt. In order to establish a safety margin and for convenience, the excitation voltage was chosen to be 10.0 volt. This voltage was supplied by the strain gauge conditioners model 2B31K manufactured by Analog Devices Inc. The 2B31K is a low cost strain gauge conditioner with the following specifications:

adjustable gain	1 to 2000 volt/volt
low frequency low pass filter	5 Hz
high frequency low pass filter	100 Hz

5.9 Software

The data acquisition software written for the experiment consists of two main parts. First there is communication software to communicate with a Dash-16 analog to digital board. This part uses calls from the Project Athena Laboratory Computer Library to initialize the Dash-16 board and to read values from the six channels. Second, the software takes 500 readings from each channel, and then takes the average and writes the data in a file together with the input from the user (tunnel speed, pitch angle and correction height). A printout for the source code is listed in appendix A.

5.10 Force and Moment Corrections

When the velocity in the tunnel test section changes, the pressure changes as well due to the Bernoulli effect. This pressure difference will result in a force on all surfaces, including and most importantly for this case, the end of the mounting shaft. The correction for the force on the end of the mounting shaft due to the pressure change is carried out as follows, (see figure 5.5):

The force ΔF from the pressure change ΔP is defined as:

$$\Delta F = \Delta P_S \cdot A_1 - \Delta P_T \cdot A_3 + \Delta P_B \cdot \left(A_2 + A_3\right) \tag{5.7}$$

Assuming the pressure ΔP_T is the same as ΔP_S the equation becomes:

$$\Delta F = \Delta P_T \cdot \left(A_1 - A_3\right) + \Delta P_B \cdot \left(A_2 + A_3\right) \tag{5.8}$$

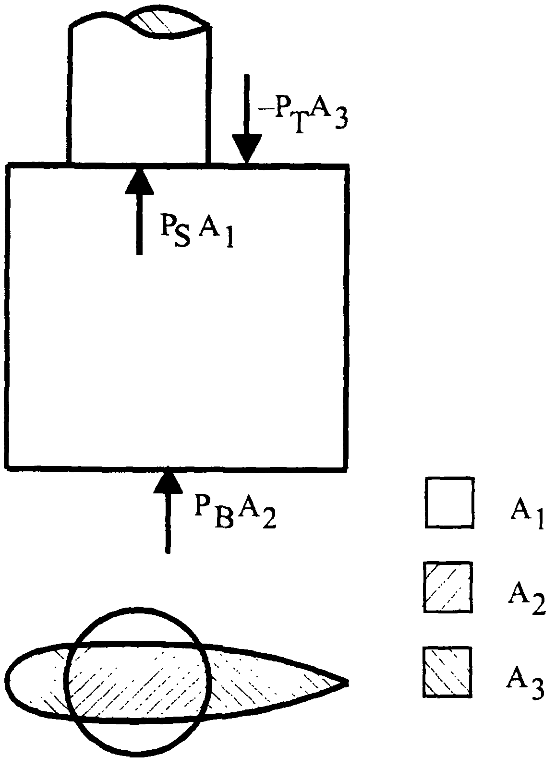


Figure 5.5 Pressure compensation on mounting shaft

The top of the cavity, where the mounting shaft slides into the vehicle, is partially sealed with a rubber gasket to minimize internal flow in the cavity. Thus the pressure in the cavity can be assumed constant. When the gap between the wall and the top of the vehicle is small, one can furthermore assume that the pressure difference between the wall and the top of the vehicle must be small. If a pressure difference were present between the wall and the vehicle, it would have to be accompanied by an acceleration component perpendicular to the wall which is unlikely taking conservation of mass and water`s incompressibility into consideration. The pressure in the gap must therefore be close to constant, so $\Delta P_T \approx \Delta P_B$.

The correction from the pressure difference on the mounting shaft is found as:

$$\Delta F = \Delta P_T \cdot \left(A_1 + A_2\right) = \Delta P_T \cdot \pi \cdot r_{SHAFT}^2 \tag{5.9}$$

A pressure tap was mounted in the end of the mounting shaft and connected to a manometer during the experiment. The manometer height h_0 was read with tunnel velocity equal to zero and the height h read again at each tunnel velocity. The pressure difference ΔP is found using equation (5.10), ρ is the density of water and g is gravity:

$$\Delta P = \rho \cdot g \cdot \left(h - h_0\right) \qquad (5.10)$$

5.12 Deflection of the dynamometer window

The pressure change due to velocity change in the tunnel is quite significant. At 24 ft/sec the manometer change ($h\text{-}h_0$) is approximately – 85 inches, equal to a pressure change of – 3.17 psi. or a distributed load of 1730 lb. on each tunnel window! This force causes the dynamometer window to deflect, resulting in loads applied on the load cells. A test run measuring the load on the dynamometer with only the round part of the mounting shaft placed flush with the wall was performed to investigate how the deflection of the window interferes with the load cells. As shown in figure 5.6, the error from the deflection of the dynamometer window is small but still significant and the data from the experiment has been corrected with the value corresponding to the fitted curve, y being the error in z force and x being the velocity.

$$y = -2\ 24 \cdot x^2 - 0\ 0235 \cdot x + 0\ 0975$$

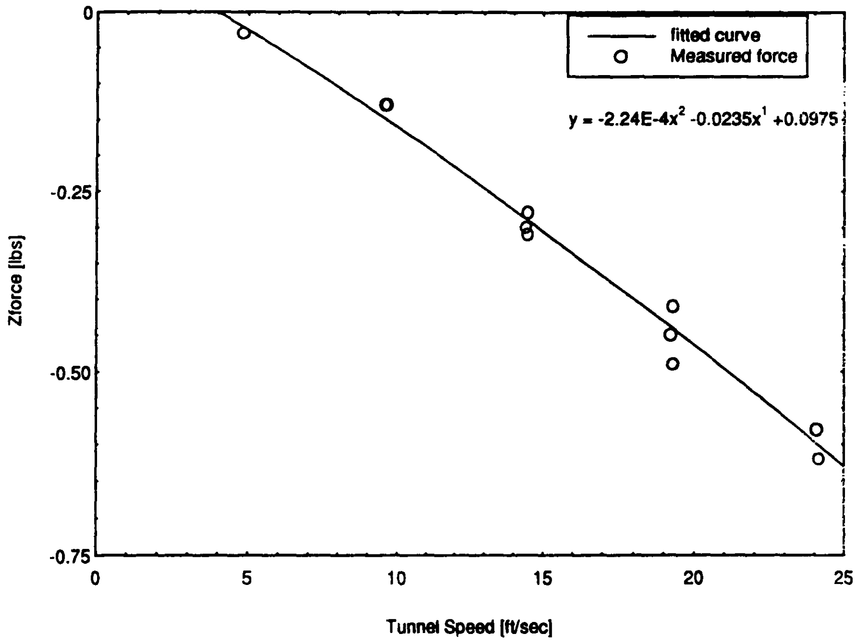


Figure 5.6: Correction of Z force vs. tunnel speed

5.12 Results

The measured results from the tests with the CSDL vehicle are shown in figures 5.7 through 5.19. The suction force coefficients and the pitch moment coefficients are plotted vs. the gap/diameter for seven different pitch angles. Where the pitch angle is equal to 0°, as shown in figures 5.7 and 5.8, the measured suction force coefficients are about 25% lower than the results from the potential flow code. This difference is quite significant, but at the same time the measured pitch moment coefficients at pitch angle = 0° are close to the potential flow code result. The fact that the moment is measured correctly and the suction force is not, indicates that the measured suction force coefficients have an erroneous offset.

There are three different factors which could effect the results, but only one of these could have the characteristics apparent in the data.

-flow disturbance from mounting shaft

- tunnel interference

- incomplete pressure correction on the mounting shaft

The mounting shaft does disturb the flow in the gap between the vehicle and the wall. If the disturbance were significant then it would be expected that the measured pitch moment would be smaller than the predicted potential flow pitch moment. A smaller pitch moment would be due to a local high pressure zone created from the stagnation point on the leading edge of the mounting shaft. Furthermore, the wake of the mounting shaft would counteract the potential flow pitch moment. For the above reasons the differences between the predictions and the measured results are not likely caused by the mounting shaft.

The far wall in water tunnel interferes with the vehicle, but looking at the curve of the suction force coefficients calculated by the potential flow code it is very unlikely that the far wall would interfere with a factor of 25 % of the suction force from the near wall. The far wall interference would increase as the vehicle move away from the near wall visa versa, and would not act as a fairly constant fraction of the force coefficient which is the case.

The pressure correction on the mounting shaft used several assumptions about the pressures on the different surfaces on the mounting shaft. The pressure correction for pressure on the end of the mounting shaft amounts in many cases to several times the suction force so a small error in the assumptions of the pressure correction scheme could cause a significant error in the suction force. The following observations are made about the error in the measured suction force coefficient:

- the error depends on the tunnel velocity squared (constant fraction of the total suction force).
- the error does not seem to interfere with the pitch moment might indicate that the erroneous force has no moment arm and is in-line with the mounting shaft and that the erroneous force is acting on the mounting shaft.

Taking the above characteristics of the error into account it is likely that the differences in the predicted and the measured suction force coefficients are due to mistaken assumption in the pressure correction on the mounting shaft. To investigate this a possibility would be to mount the shaft and the model in the tunnel without them being connected, supporting the model by other means and then measure the forces on the shaft to compare with the pressure corrections made in this experiment.

At positive pitch angles, when the bow of the vehicle is turned away from the wall, both the measured suction force coefficients and the pitch moment coefficients are smaller than the potential flow code results. At higher pitch angles the sign of the measured suction force is reversed, thus the vehicle is forced away from the wall. The measured pitch moment decreases to approximately 50% of the potential flow code result as the pitch angle increases. The explanation of the differences is to be found in the viscous effects. When the vehicle moves through the fluid at large pitch angle, the flow will separate and create a low pressure wake behind the vehicle. This low pressure wake will be on the far side from the wall when the vehicle bow is turned away from the wall and the vehicle will experience a viscous suction force away from the wall. This viscous force counteracts the potential suction force and the resulting suction force decreases as the pitch angle increases. The pitch moment coefficient decreases as the pitch angle and the viscous wake increases. This indicates that the low pressure wake is acting stronger on the aft part of the vehicle than on the bow and therefore counteracts the potential pitch moment.

At negative pitch angles the measured suction force coefficients are increasingly larger than the potential flow code results as the pitch angle increases. This agrees with the above explanation. In the case of negative pitch angles a low pressure wake is located on the near side of the vehicle towards the wall and the viscous suction force acts in the same direction as the potential flow suction force. The measured pitch moment coefficients are smaller in cases with negative pitch angles. In this case the viscous suction force acts on the aft part of the vehicle and thus counteracts the potential flow suction force which acts on the forward end of the vehicle, where there is the smallest gap.

The measured results from the experiment with the WHOI model is shown in figure 5.20. The poor signal to noise ratio is apparent, the measured suction force coefficients at each vehicle position varies over an interval of approximately a third of the largest measured value. The average values for each vehicle position are plotted for convenience. The data show a trend of increasing suction force as the gap/diameter decreases. the signal to noise ratio appears to be quite low and the advantages of a larger vehicle is apparent.

Figure 5.7:

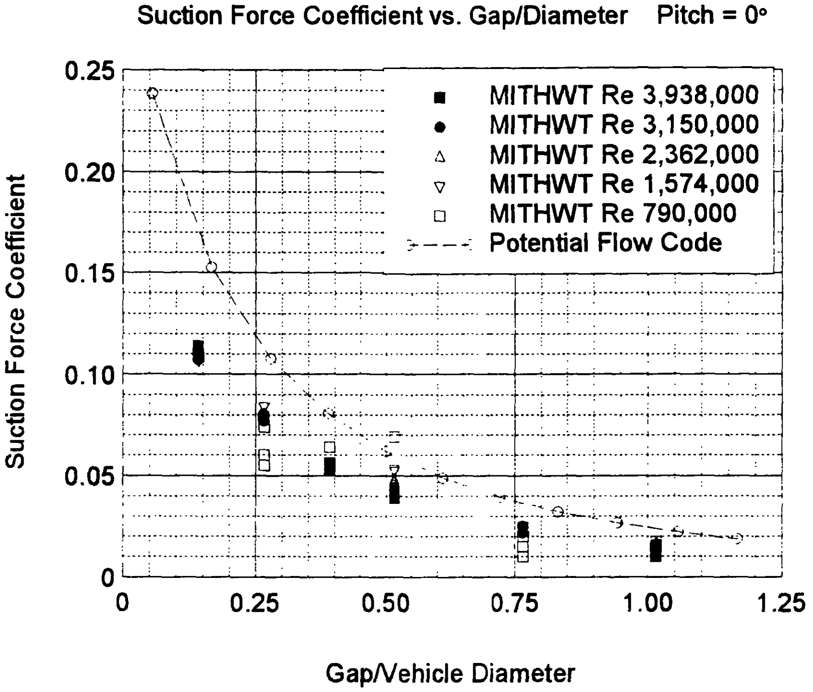


Figure 5.8:

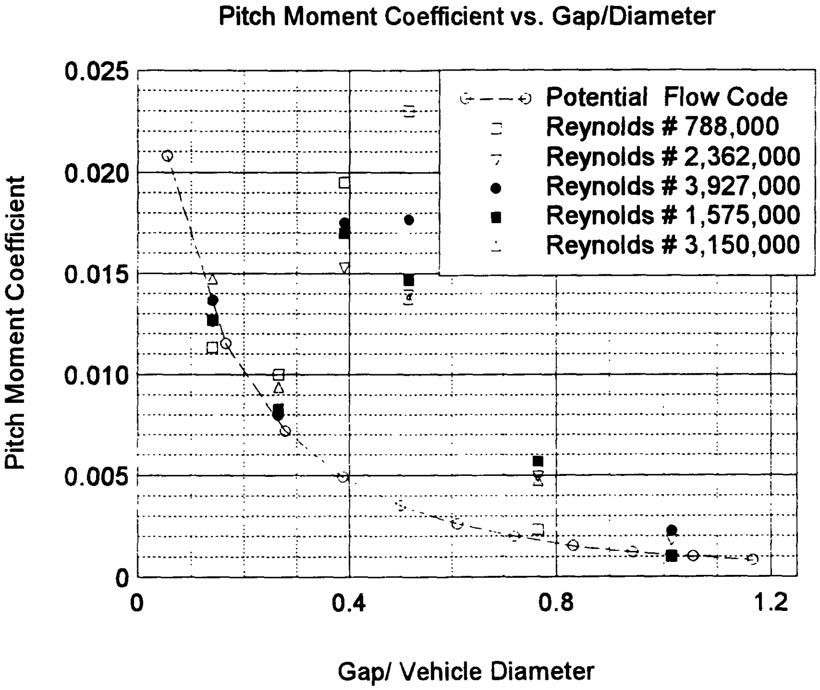


Figure 5.9:

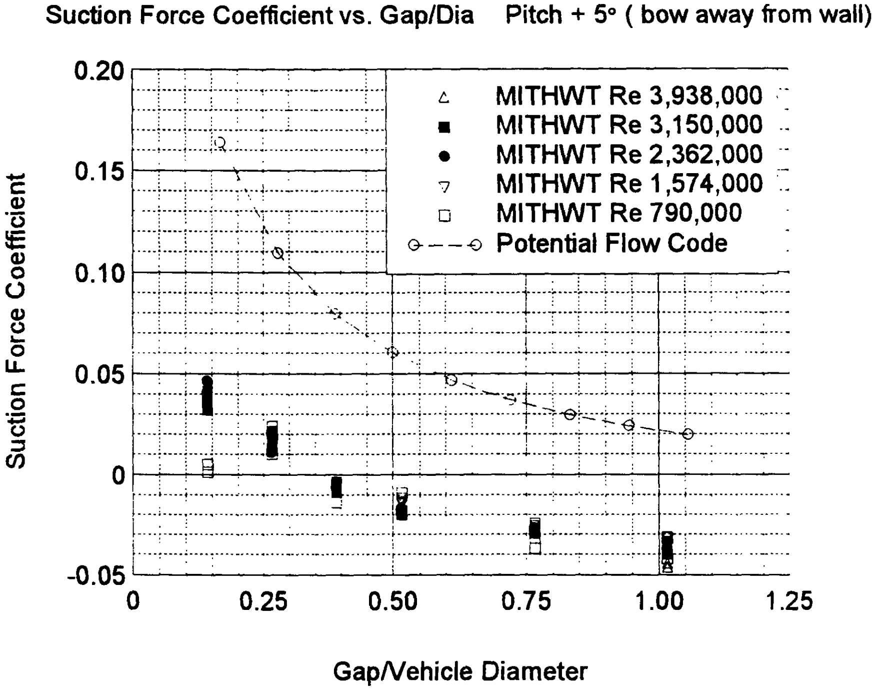


Figure 5.10:

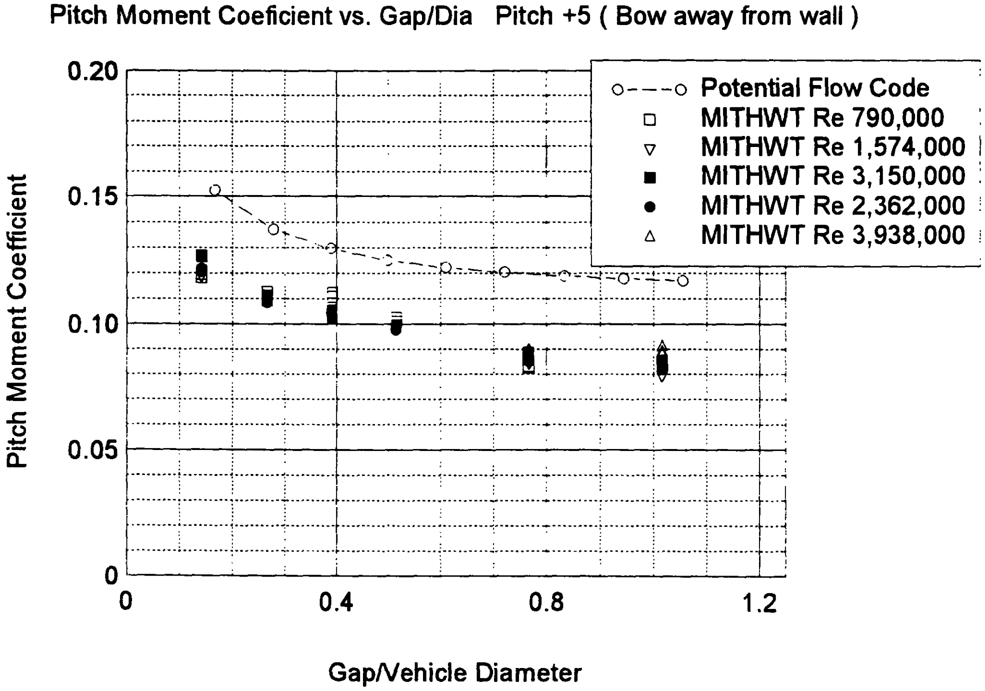


Figure 5.11:

Suction Force Coefficient vs. Gap/Diameter Pitch +10° (Bow away from wall)

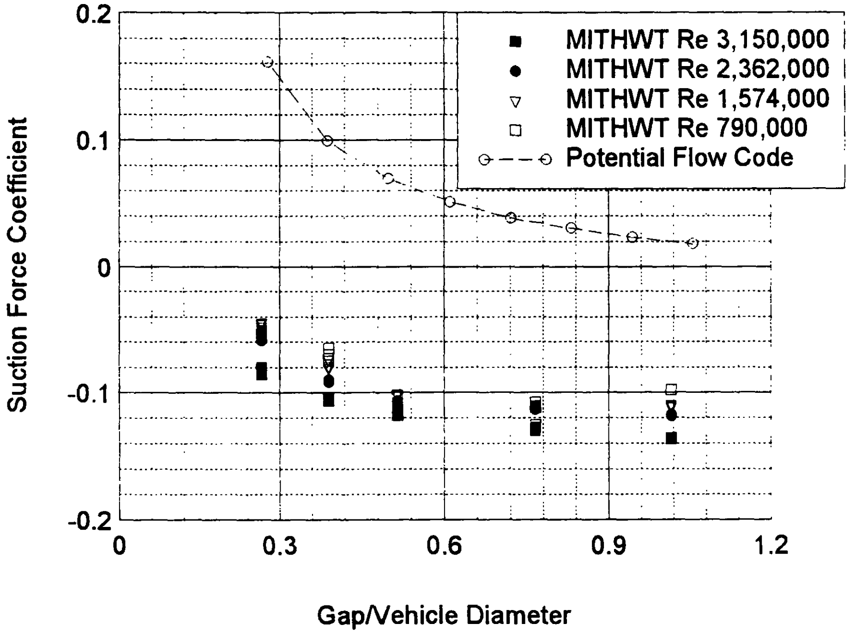


Figure 5.12:

Pitch Moment Coefficient vs. Gap/Diameter Pitch +10° (Bow away from wall)

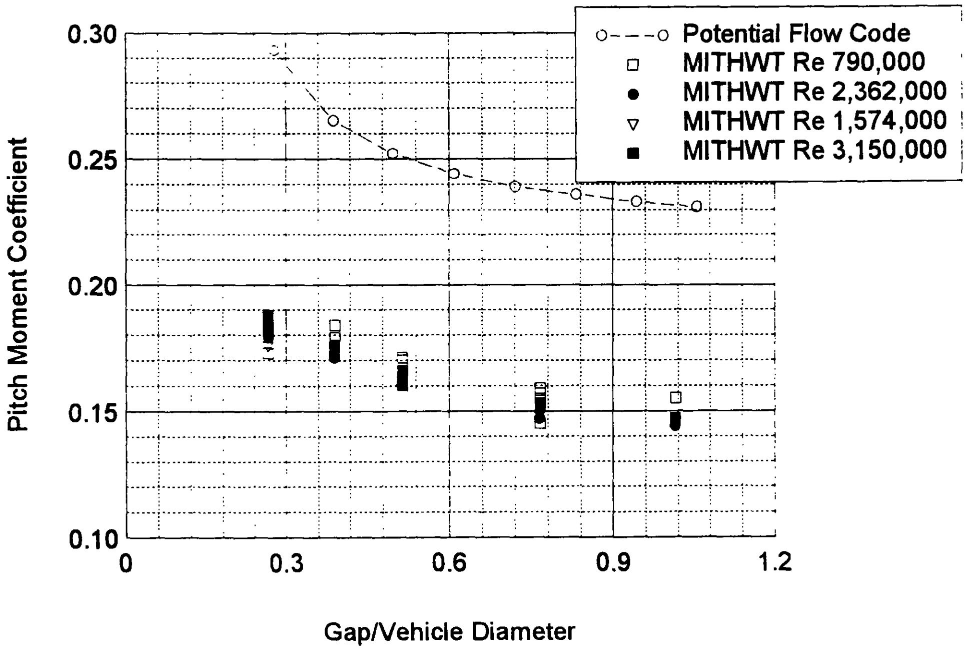


Figure 5.13:

Suction Force Coefficient vs. Gap/Dia Pitch + 15 ° (bow away from wall)

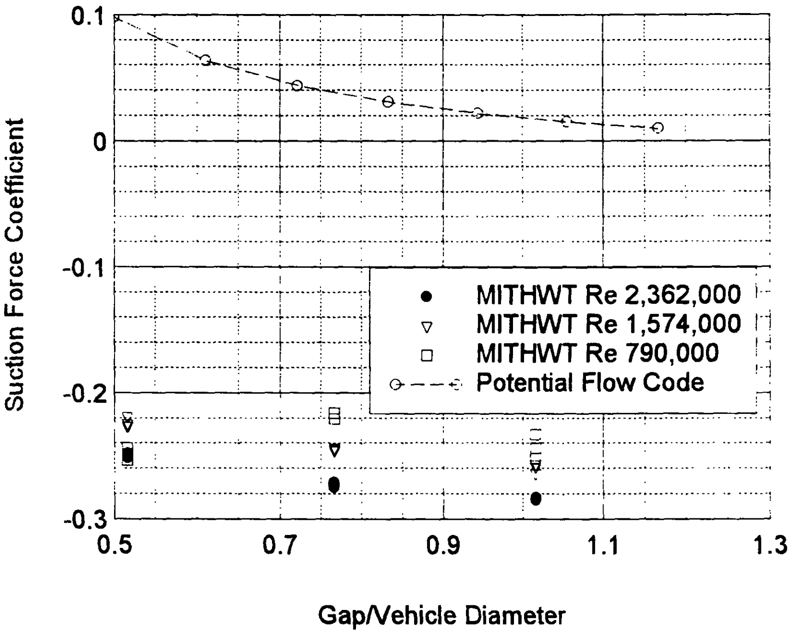


Figure 5.14:

Pitch Moment Coefficient vs. Gap/Diameter Pitch +15° (Bow Away from wall)

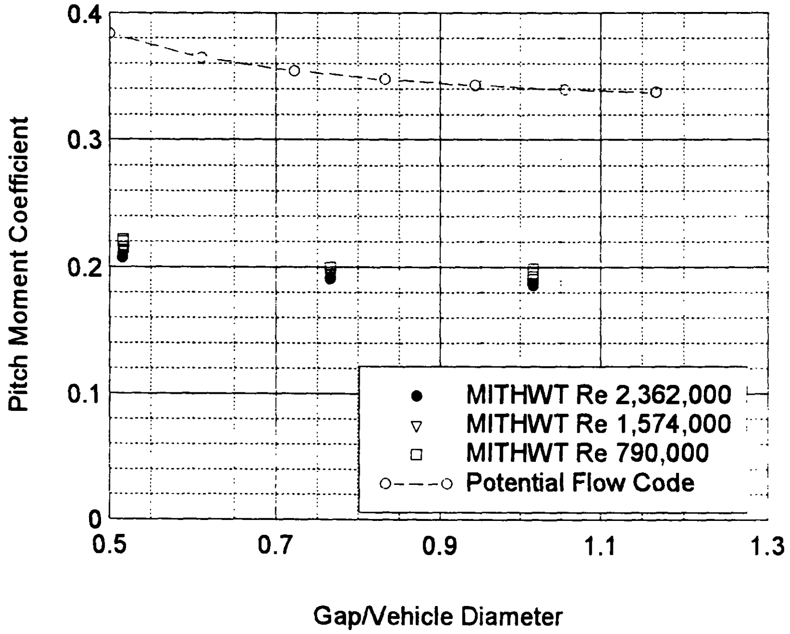


Figure 5.15:

Suction Force Coefficient vs. Gap/Diameter Pitch -5° (Bow Towards Wall)

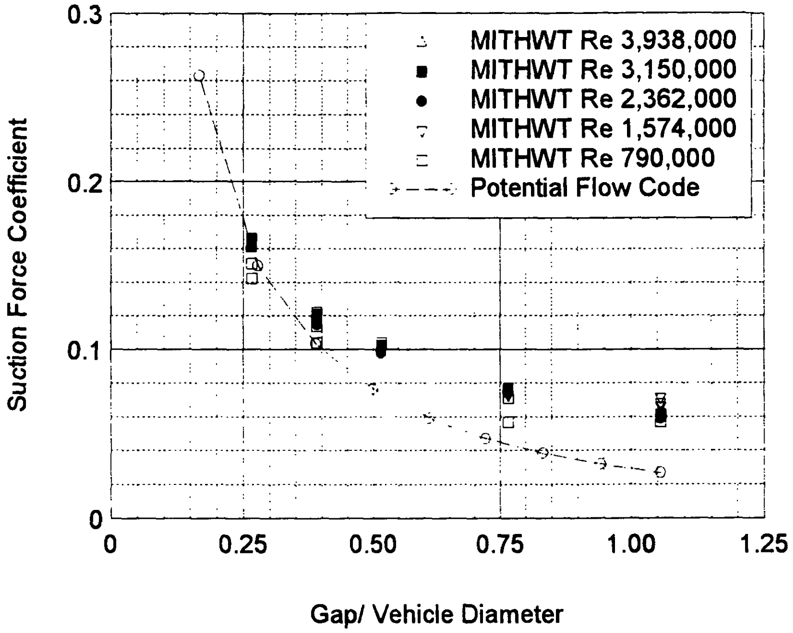


Figure 5.15:

Pitch Moment Coefficient vs. Gap/Diameter Pitch -5° (Bow towards Wall)

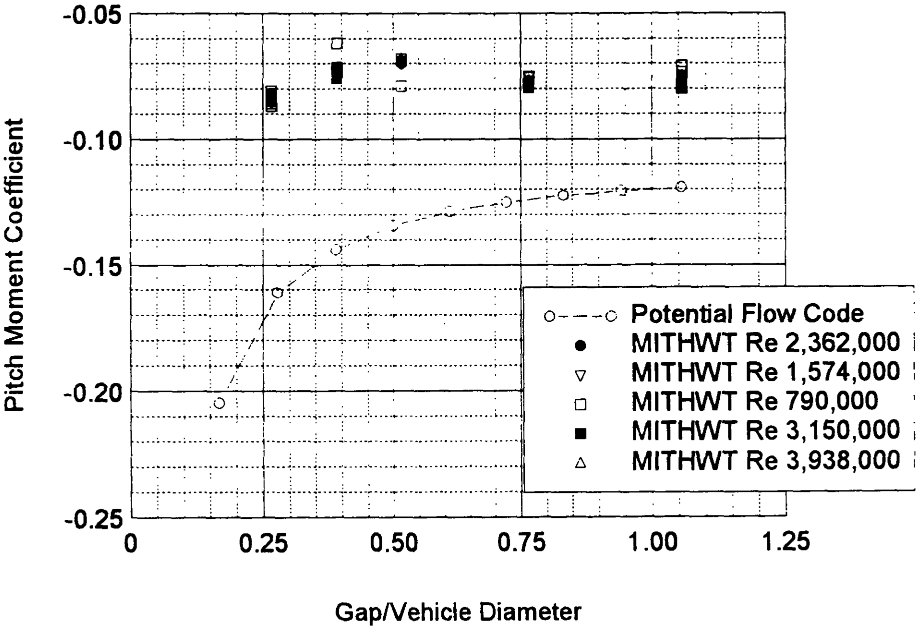


Figure 5.16:

Suction Force Coefficient vs. Gap/Diameter Pitch -10° (Bow Towards Wall)

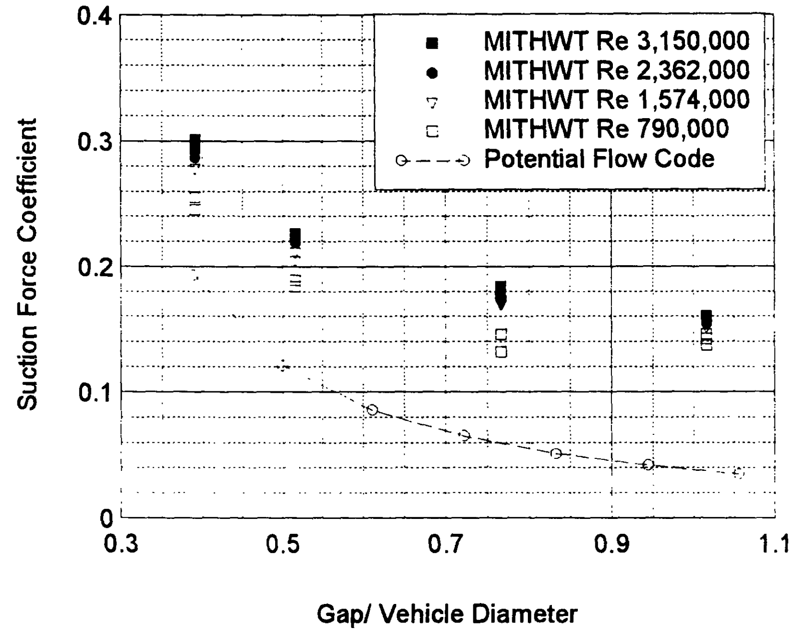


Figure 5.17:

Pitch Moment Coefficient vs. Gap/Diameter Pitch -10° (Bow Towards Wall)

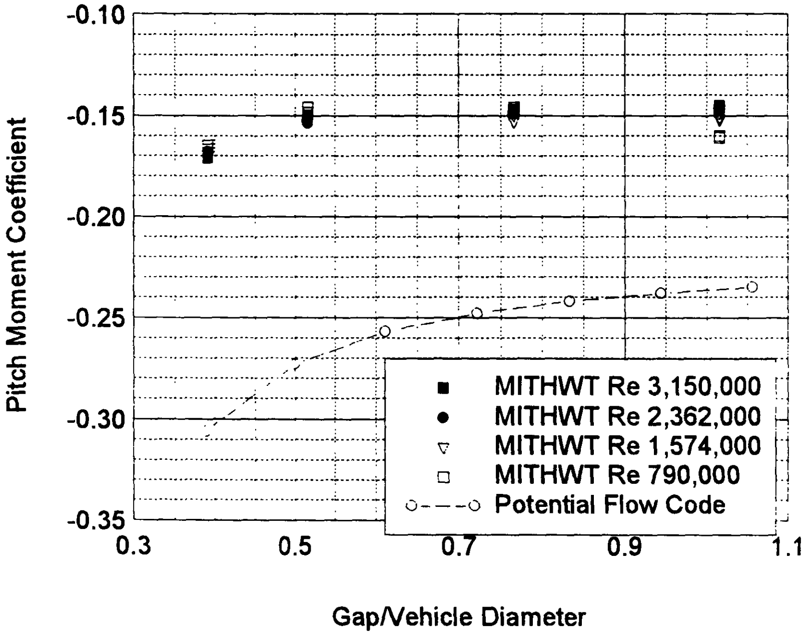


Figure 5.18:

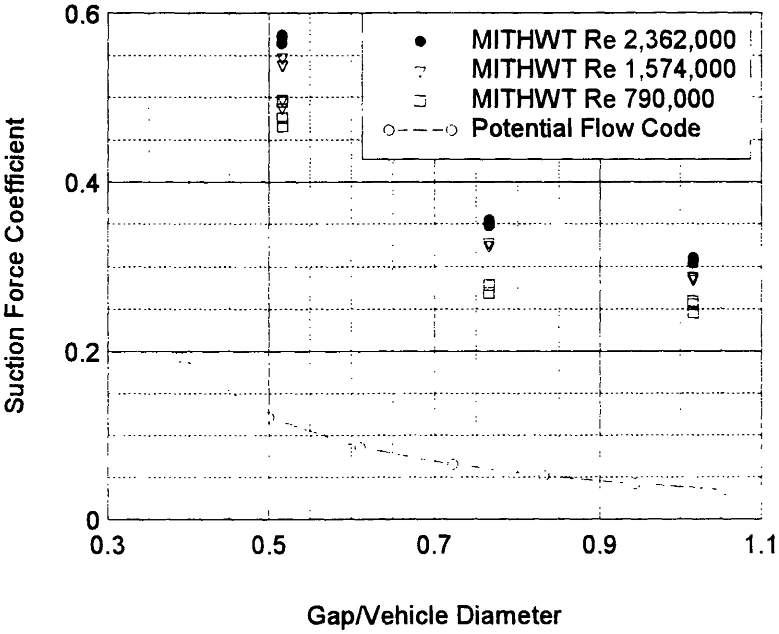
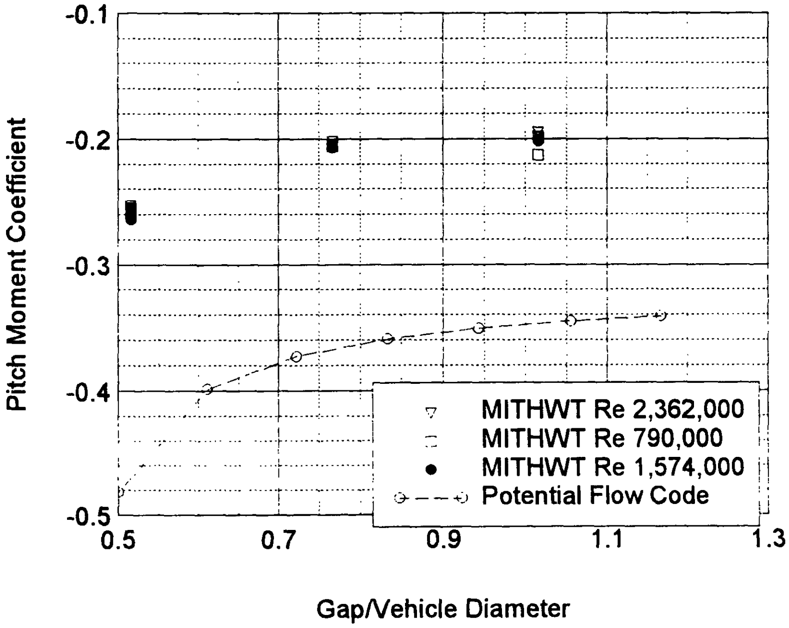


Figure 5.19:



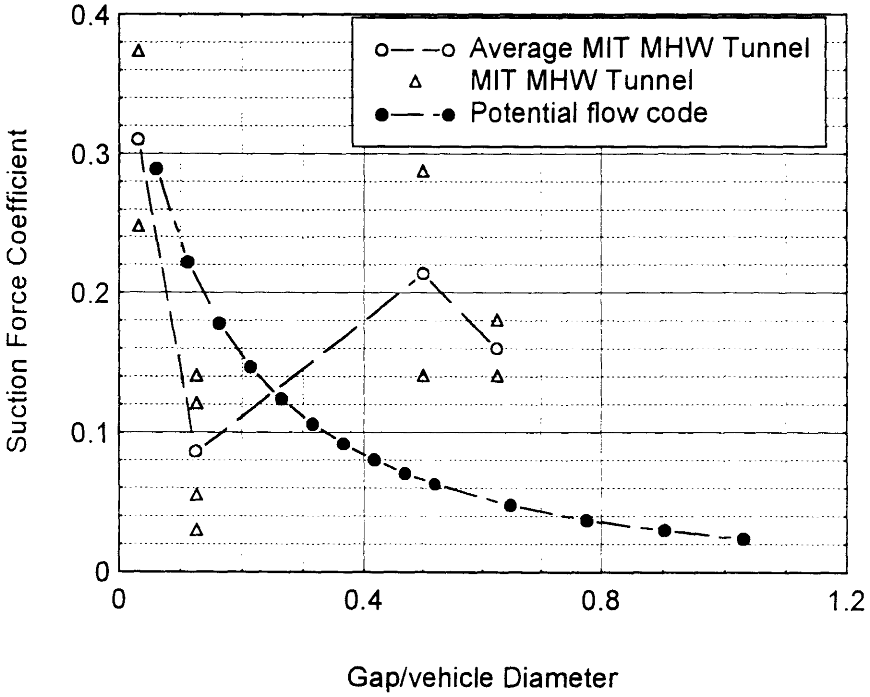


Figure 5.20: suction force on WHOI vehicle

Chapter 6

Discussion

In this thesis a potential based boundary integral element method computer code was developed for calculating forces and moments on a vehicle near a solid boundary. The code uses a combined source and dipole formulation to solve for the perturbation potential.

The code converges very rapidly to the theoretical result for the tangential velocity on a sphere. The calculated suction force on a sphere near a wall agrees very well with the analytical approximation for conditions under which the analytic approximation is accurate. These are for the gap between the sphere and the wall being greater than the radius of the sphere.

Comparing the potential based code with slender body theory it was found that the results agreed when the test vehicle is a very slender spheroid with length to diameter ratio of 40. The results disagree when the vehicle is a spheroid with a length to diameter of 8.0, or of blunter shape. In these latter cases the conditions of slender body theory are violated to the extend that the theory has significant inaccuracies. This defect is remedied by the BIEM code.

The potential based code was also used to investigate the suction force and pitch moment on two similar AUV's. The results showed that the suction force is mostly dependent on the "bulkiness" of the vehicle. In other words the amount of area of the body that gets near the wall effects the suction force. The pitch moment depends on the fore-and-aft symmetry of the vehicle, so if the vehicle is symmetrical there will be no pitch moment. The code showed that if the vehicle is bulkier in the bow there will be a pitch moment trying to force the bow towards the wall.

Two experiments were conducted in the MIT Hydrodynamics Water Tunnel using models of the CSDL and WHOI vehicles. These tests were conducted to assess the real fluid effect on the flow around the vehicles and thereby get a measure of the applicability of the numerical results in real life situations. The obtained results for the pitch moment showed good correlation with the numerical results for vehicles with no pitch but the measured suction forces were approximately 30 % lower than the numerical results. It is likely that the differences in the predicted and the measured suction force coefficients are due to mistaken assumption in the pressure correction on the mounting shaft. An approach to investigate this theory would be to mount the shaft and the model in the tunnel without having them connected. Measuring the forces on the shaft alone by supporting the model by other means and compare the result before and after the pressure correction.

The tests showed furthermore, that the pitch angle has great influence on both the suction force and the pitch moment. It was discovered that when the vehicle is pitched at an angle with the flow, then the flow separates on the downstream side of the vehicle. This separation creates a low-pressure wake resulting in a force on the vehicle in the same direction as the wake. The wake is most developed towards the tail of the vehicle resulting in a pitch moment trying to turn the vehicle back in line with the flow. The consequences for these phenomena are that when the vehicle is turned with the bow towards the wall the force from the wake works in the same direction as the potential flow suction force. In the case where the vehicle bow is turned away from the wall, the wake force works in the opposite direction as the potential flow suction force. In both cases the pitch moment from the wake will counteract the potential flow pitch moment.

Bibliography

[1] R. W. Clark, "A new iterative matrix solution procedure of three-dimensional panel methods", AIAA 23rd Aerospace Sciences Meeting, page 1-6 ,Reno, Nevada, January 14-17 1985.

[2] J. L. Hess and A. M. O. Smith, "Calculation of non-lifting potential flow about arbitrary bodies", Report No. E.S. 40622, DOUGLAS Aircraft Company Inc., Long Beach CA, 1962.

[3] J.L. Hess, "Calculation of potential flow about arbitrary three-dimensional lifting bodies", Technical Report MDC J5679-01, McDonnell Douglas, October 1972.

[4] J. T. Lee, "A potential Based Panel Method for the Analysis of Marine Propellers in Steady Flow", Report No. 87-13, Massachusetts Institute of Technology, 1987

[5] L.M. Milne Thomson, "Theoretical Hydrodynamics", 1960

[6] L. Morini and C. C. Kuo, "Subsonic potential aerodynamic for complex configurations: A general theory", AIAA J., 12(2):191-197, February 1974.

[7] J. N. Newman, Marine Hydrodynamics, The MIT Press, Cambridge Massachusetts, 1977

[8] J.N. Newman, "Distribution of sources and normal dipoles over quadrilateral panels", Department of Ocean Engineering , Massachusetts Institute of Technology, 1985

[9] H. S. Ölmez, "Numerical Evaluation of nonlinear energy transfer to short gravity waves in the presence of long waves", Sc.D. Thesis, Massachusetts Institute of Technology, 1991

Appendix A

```
$DEBUG
    PROGRAM TORDATA
    CHARACTER FNAME*12,CH1*1, CH2*1
    DIMENSION IVAL(11000), VAL(8),JVAL(8)
    M      = 1
    N      = 6
    NCH    = 6
    IG     = 1
    NREC   = 0
    FREQ   = 100.
    PER    = 1.0/FREQ
7    WRITE (*,8)
8   FORMAT(1X,' TYPE NAME OF FILE TO STORE RESULTS ')
    WRITE (*,'(A,$)')' USE EXTENSION .RAW OR .ZRO '
    READ (*,'(A)') FNAME
    OPEN (3, FILE = FNAME, STATUS= 'NEW',ERR = 700)
    CLOSE (3)
10   WRITE(*,'(A,$)')' TYPE A <RET> TO ACQUIRE DATA, Q <RET> TO QUIT '
    READ (*,'(A)') CH1
    IF ((CH1.EQ. 'A') .OR. (CH1 .EQ. 'A')) THEN
        GO TO 100
    ELSEIF ((CH1 .EQ. 'Q') .OR. (CH1 .EQ. 'Q')) THEN
       WRITE (*,'(A,$)')' ARE YOU SURE YOU WANT TO QUIT? (Y/N)  '
       READ (*,'(A)') CH2
       IF ((CH2 .EQ. 'Y') .OR. (CH2 .EQ. 'Y')) GO TO 900
       GO TO 10
    ELSE
        GO TO 10
    ENDIF
100  DO 90 J = 1,6
    VAL(J) = 0.0
90   JVAL(J) = 0
    CALL MBOPEN
    CALL ATODNOW(M,IG,JV)
    CALL SETCLOCK (PER,0)
    CALL MCBATOD(M,N,IG,500,IVAL)
    CALL GETDAT(IYR,IMON,IDAY)
    CALL GETTIM(IHR,IMIN,ISEC,I100TH)

    DO 110 I = 1,500
    DO 120 J = 1,6
120  VAL(J) = VAL(J) + IVAL( 6*(I-1) + J)
110  CONTINUE
```

```
    WRITE (*,*) (VAL(J),J = 1,6)
    DO 130 J = 1,6
130  JVAL(J) = NINT(VAL(J)/500.)
    NRP=NREC+1
115  WRITE (*,'(A,$)')' TYPE TUNNEL(RPM) & MANO HT '
    READ (*,*,ERR=115) IRPM, H
    WRITE (*,140) NRP,(JVAL(J),J = 1,6), IRPM,H,IMON,IDAY,
   1  IYR,IHR,IMIN,ISEC
140  FORMAT (1X,I3,6I6,1X,I3,1X,F5.1,2X,I2.'/',I2,'/',I4,1X,
   1  I3,':',I2,':',I2)
145     WRITE (*,'(A,$)')' DO YOU WANT TO FILE THESE DATA? (Y/N)  '
    READ (*,'(A)') CH1
    IF ( (CH1 .EQ. 'Y') .OR. (CH1 .EQ. 'Y')) THEN
     OPEN (3, FILE = FNAME,STATUS='OLD')
     IF ( NREC .LT. 1) GO TO 150
    DO 155 I = 1, NREC
155   READ (3,*,END=150) IJUNK
150  CONTINUE
     NREC = NREC + 1
     WRITE (3,140) NREC,(JVAL(J),J = 1,6),IRPM,H,
   1 IMON,IDAY,IYR,IHR,IMIN,ISEC
     CLOSE (3)
    ELSEIF  ((CH1.EQ.'N') .OR. (CH1.EQ.'N')) THEN
      GO TO 10
    ELSE
     GO TO 145
    ENDIF
    GO TO 10
700  WRITE(*,*)
    WRITE(*,*)'FILE ALREADY EXISTS, PLEASE TRY AGAIN.....'
    WRITE(*,*)
    GO TO 7
900     STOP
    END
```

Appendix B

```
$DEBUG
    PROGRAM SJCAL
    CHARACTER FNAME*20, GNAME*20
    DIMENSION V(9,6),F(9,6),B(10,10),X(10),IL(10),A(6,6)
    WRITE (*,'(A,$)')' TYPE NAME OF INPUT FILE  '
    READ (*,'(A)') FNAME
    OPEN (2, FILE=FNAME,STATUS='OLD')
    WRITE (*,'(A,$)')' TYPE NAME OF OUTPUT FILE  '
    READ (*,'(A)') GNAME
    OPEN (3, FILE=GNAME,STATUS='UNKNOWN')
    OPEN (4, FILE = 'SORJUNK', STATUS='UNKNOWN')
    DO 100 NPULL= 1,9
    DO 100 I = 1,6
100  READ(2,*) V(NPULL,I), F(NPULL,I)
    DO 200 NROWC = 1,6
    DO 260 NRM = 1,9
    DO 250 NCM = 1,6
250  B(NRM,NCM) = V(NRM,NCM)
    B(NRM,7) = F(NRM,NROWC)
C    WRITE (3,'(7F11.5)') (B(NRM,J), J = 1,7)

260  CONTINUE
    CALL GLSQSJ(B,X,IL,9,6,ALPHA,0.00000,0.00000)
    DO 270 I = 1,6
270  A(NROWC,I) = X(I)
    WRITE (3,'(6F12.5)') (X(I), I = 1,6)
200  CONTINUE

    DO 400 NPULL = 1,9
    DO 350 NF = 1,6
    FC = 0.0
    DO 300 NC = 1,6
300  FC = FC + A(NF,NC) * V(NPULL,NC)
    WRITE (4,'(2E14.4)') F(NPULL,NF),FC
350   WRITE (*,*) F(NPULL,NF), FC
400  CONTINUE
    END
```

```
      SUBROUTINE GLSQSJ(A,X,IL,N,M,ALPHA,E1,E2)
      DIMENSION A(10,10),X(10),IL(10)
      MM=M+1
      LL=1
      DO 60 J=1,MM
60     IL(J)=0
      I=1
      DO 3 K=1,MM
      II=I+1
      DO 4 J=II,N
      IF (ABS(A(J,K))-E1) 4,4,6
6      T1=SQRT((A(J,K))**2+(A(I,K))**2)
      S=A(J,K)/T1
      C=A(I,K)/T1
      DO 5 L=K,MM
      T2=C*A(I,L)+S*A(J,L)
      A(J,L)=-S*A(I,L)+C*A(J,L)
5      A(I,L)=T2
      LL=LL+1
4      CONTINUE
      IF (ABS(A(I,K))-E2) 3,3,8
8      IL(K)=I
      I=I+1
3      CONTINUE
      X(MM)=-1.0
      II=M
      DO 35 I=1,M
35      X(I)=0.0
      DO 30 J=1,M
      IF (IL(II)) 30,30,31
31      S=0.0
      LL=II+1
      I=IL(II)
      DO 32 K=LL,MM
32     S=S+A(I,K)*X(K)
      X(II)=-S/A(I,II)
30     II=II-1
      IF (IL(MM)) 50,51,50
51      ALPHA=0.0
      GO TO 52
50     I=IL(MM)
      ALPHA=A(I,MM)
52     RETURN
      END
```

Appendix C

```
      SUBROUTINE PRESSURE(NPANEL,RHO,QRM,BLEN,CG,SIG,NT,NX,I7)
      INCLUDE "PARAMETER.INC"
      COMMON /C1/ XP(MAXDIM,4),YP(MAXDIM,4),ZP(MAXDIM,4),PI,
     & XC(MAXDIM),YC(MAXDIM),ZC(MAXDIM),V(3)
      COMMON /C2/ DC(3,3),ETA(5),QSI(5),X0,Y0,Z0,DA(MAXDIM),
     & VNORMAL(MAXDIM,3),VELABS(MAXDIM)
      DIMENSION FORCE(6),VN(3),R(3),RCN(3),CG(3),SIG(MAXDIM)
C-------------------------------------------------------
C   INITIALIZE FORCE VECTOR
C-------------------------------------------------------
      DO 45 I=1,6
       FORCE(I)= 0.0
 45    CONTINUE
       AREA   = 0.0
C-------------------------------------------------------
C    STAGPRESS IS THE VELOCITY SQUARED
C-------------------------------------------------------
       STAGPRES = V(1)*V(1)+V(2)*V(2)+V(3)*V(3)
      OPEN(22,FILE='CP.DAT',STATUS='UNKNOWN')
      WRITE(22,*) 'VARIABLES = "X", "Y", "Z", "CP", "POT","ABSVEL"'
       DO 50 J = 1, NPANEL
        R(1)=XC(J)-CG(1)
        R(2)=YC(J)-CG(2)
        R(3)=ZC(J)-CG(3)
C-------------------------------------------------------
C   VN IS THE NORMAL VECTOR
C-------------------------------------------------------
        VN(1)=VNORMAL(J,1)
        VN(2)=VNORMAL(J,2)
        VN(3)=VNORMAL(J,3)
        FORCEMAG=-0.5*RHO*(STAGPRES-(VELABS(J)*VELABS(J)))*DA(J)
        CP = 1.0-VELABS(J)*VELABS(J)/STAGPRES
        WRITE(22,*) XC(J),YC(J),ZC(J),CP,SIG(J),VELABS(J)
        VN(1)=VN(1)*FORCEMAG
        VN(2)=VN(2)*FORCEMAG
        VN(3)=VN(3)*FORCEMAG
        FORCE(1) = FORCE(1)+VN(1)
        FORCE(2) = FORCE(2)+VN(2)
        FORCE(3) = FORCE(3)+VN(3)
        CALL CROSS(R,VN,RCN)
        FORCE(4) = FORCE(4)+RCN(1)
        FORCE(5) = FORCE(5)+RCN(2)
        FORCE(6) = FORCE(6)+RCN(3)
        AREA=AREA+DA(J)
 50   CONTINUE
      WRITE(I7,123)AREA

       WRITE(I7,*) ' FORCES AND MOMENTS:'
       WRITE(I7,*)
       WRITE(I7,*)
       WRITE(I7,*)'FORCE(1),  FORCE(2),  FORCE(3) :'
       WRITE(I7,14)FORCE(1),FORCE(2),FORCE(3)
       WRITE(I7,*)'MOMENT(1),  MOMENT(2),  MOMENT(3)  :'
       WRITE(I7,14)FORCE(4),FORCE(5),FORCE(6)

      DO 199 J=1,3
        FORCE(J)=FORCE(J)/QRM
        FORCE(J+3)=FORCE(J+3)/(QRM*BLEN)
 199  CONTINUE
      WRITE(I7,*) ' FORCE AND MOMENT COEFFICIENTS :'
      WRITE(I7,*)
      WRITE(I7,*) ' FORCE COEFFICIENTS :( F/0.5*RHO*VEL^2*FRONTAL AREA)'
      WRITE(I7,14)FORCE(1),FORCE(2),FORCE(3)
      WRITE(I7,*)
      WRITE(I7,*) ' MOMENT COEFFICIENTS :( MOMENT/0.5*RHO*VEL^2*
     & FRONTAL_AREA* LENGTH)'
      WRITE(I7,*)
      WRITE(I7,*)'MOMENT(1),MOMENT(2),MOMENT(3)'
      WRITE(I7,14)FORCE(4),FORCE(5),FORCE(6)
 14    FORMAT(1X,3F9.4)
 121    FORMAT(1X,5F8.3)
 122  FORMAT(1X,I5,3F12.5)
 123  FORMAT(1X,' SURFACE AREA  : ',F8.3)


      RETURN
      END


      SUBROUTINE CROSS(A,B,C)
C--------------------------------------------------------------------
C   CALCULATES CROSS PRODUCT OF TWO VECTORS
C--------------------------------------------------------------------
      DIMENSION A(3),B(3),C(3)
      C(1)=A(2)*B(3)-A(3)*B(2)
      C(2)=B(1)*A(3)-A(1)*B(3)
      C(3)=A(1)*B(2)-A(2)*B(1)
```

```
RETURN
END
```

Appendix D

```
      SUBROUTINE VELO(NPANEL,B,NT,NX)
      INCLUDE "PARAMETER.INC"
      COMMON /C1/ XP(MAXDIM,4),YP(MAXDIM,4),ZP(MAXDIM,4),PI,
     & XC(MAXDIM),YC(MAXDIM),ZC(MAXDIM),V(3)
      COMMON /C2/ DC(3,3),ETA(5),QSI(5),X0,Y0,Z0,DA(MAXDIM),
     & VNORMAL(MAXDIM,3),VELABS(MAXDIM),TRANS(MAXDIM,6)
      COMMON /VELOCI/ VEL(MAXDIM,3),VECTOR1(MAXDIM,3),
     & VECTOR2(MAXDIM,3)
      DIMENSION B(MAXDIM)
C
C---------------------------------------------------------
C   OUTER LOOP
C---------------------------------------------------------
      DO 801 J=1,NX
          DO 601 I=1,NT
      NCOUNT=(J-1)*NT+I
C---------------------------------------------------------
C   PANELS IN THE FIRST RING
C---------------------------------------------------------
        IF ((J.EQ.1).AND.(2*NCOUNT.LE.NT))THEN
          NEXT = NCOUNT+NT
          NPREVIOUS=NCOUNT+NT/2
          ELSEIF ((J.EQ.1).AND.(2*NCOUNT.GT.NT))THEN
             NEXT = NCOUNT+NT
             NPREVIOUS=NCOUNT-NT/2
          ENDIF
C---------------------------------------------------------
C   PANELS IN THE LAST RING
C---------------------------------------------------------
        IF ((J.EQ.NX).AND.(NCOUNT.LE.(NPANEL-NT/2)))THEN
          NEXT = NCOUNT+NT/2
          NPREVIOUS=NCOUNT-NT
          ELSEIF ((J.EQ.NX).AND.(NCOUNT.GT.(NPANEL-NT/2)))THEN
             NEXT = NCOUNT-NT/2
             NPREVIOUS=NCOUNT-NT
             ENDIF
C----------------------------------------------------------
C   DOUBLE SIDED DIFFERENTIATION PANELS SURROUNDED BY OTHER
C   PANELS ON ALL SIDES
C----------------------------------------------------------
       IF((J.GT.1).AND.(J.LT.NX))THEN
         NEXT = NCOUNT+NT
         NPREVIOUS = NCOUNT-NT
      ENDIF
C-----------------------------------------------------------
C     VELOCITY IN DIRECTION 1
C-----------------------------------------------------------
      DELPHI1 =  B(NCOUNT)-B(NPREVIOUS)
      DELPHI2 =  B(NEXT)-B(NCOUNT)
      VECX1   = XC(NCOUNT)-XC(NPREVIOUS)
      VECY1   = YC(NCOUNT)-YC(NPREVIOUS)
      VECZ1   = ZC(NCOUNT)-ZC(NPREVIOUS)
      VEC1    = SQRT(VECX1*VECX1+VECY1*VECY1+VECZ1*VECZ1)
      VECTOR1(NCOUNT,1) = XC(NEXT)-XC(NPREVIOUS)
      VECTOR1(NCOUNT,2) = YC(NEXT)-YC(NPREVIOUS)
      VECTOR1(NCOUNT,3) = ZC(NEXT)-ZC(NPREVIOUS)
      VEL(NCOUNT.1) = SQRT((VECTOR1(NCOUNT.1)*VECTOR1(NCOUNT,1))
     &     + (VECTOR1(NCOUNT,2)*VECTOR1(NCOUNT,2))
     &      +(VECTOR1(NCOUNT,3)*VECTOR1(NCOUNT,3)))
C-----------------------------------------------------------
C     VECL(NCOUNT.1) IS SUM OF VEC1 AND VEC2
C-----------------------------------------------------------
      VECX2  = VECTOR1(NCOUNT.1)-VECX1
      VECY2  = VECTOR1(NCOUNT,2)-VECY1
      VECZ2  = VECTOR1(NCOUNT,3)-VECZ1
      VEC2   = SQRT(VECX2*VECX2+VECY2*VECY2+VECZ2*VECZ2)
C-----------------------------------------------------------
C     VECTOR1 AS UNIT VECTOR
C-----------------------------------------------------------
      VECTOR1(NCOUNT,1) = VECTOR1(NCOUNT,1)/VEL(NCOUNT,1)
      VECTOR1(NCOUNT,2) = VECTOR1(NCOUNT,2)/VEL(NCOUNT,1)
      VECTOR1(NCOUNT,3) = VECTOR1(NCOUNT,3)/VEL(NCOUNT,1)
      VEC2  = VEC1+VEC2
C-----------------------------------------------------------
C      QUADRATIC AX^2+BX+C  ARE CALCULATED
C-----------------------------------------------------------

      DELX1  = VEC1
      DELX2  = VEC2 - VEC1
      DELSQ1 = VEC1*VEC1
      DELSQ2 = (VEC2*VEC2)-(VEC1*VEC1)
      N=(NCOUNT)/NT
      CONSTB = ((DELPHI2*DELSQ1)-(DELPHI1*DELSQ2))
     &   /((DELSQ1*DELX2)-(DELSQ2*DELX1))
      CONSTA = (DELPHI1-CONSTB*DELX1)/DELSQ1
      VEL(NCOUNT,1)     = CONSTB+2.*CONSTA*VEC1

C-----------------------------------------------------------
```

```
C    VELOCITY IN DIRECTION 2
C-----------------------------------------------------------
      M=(NCOUNT-1)/NT
      N=(NCOUNT)/NT
      IF (NCOUNT .EQ. N*NT) THEN
        NLEFT =NCOUNT-NT+1
        NRIGHT = NCOUNT-1
        ELSEIF ((NCOUNT-1) .EQ. M*NT) THEN
        NLEFT =NCOUNT+1
        NRIGHT = (NCOUNT-1)+NT
      ELSE
        NLEFT =NCOUNT+1
        NRIGHT = NCOUNT-1
      ENDIF
      DELPHI1 = B(NCOUNT)-B(NLEFT)
      DELPHI2 = B(NRIGHT)-B(NCOUNT)
      VECX1 = XC(NCOUNT)-XC(NLEFT)
      VECY1 = YC(NCOUNT)-YC(NLEFT)
      VECZ1 = ZC(NCOUNT)-ZC(NLEFT)
      VEC1  = SQRT(VECX1*VECX1+VECY1*VECY1+VECZ1*VECZ1)
      VECTOR2(NCOUNT,1) = XC(NRIGHT)-XC(NLEFT)
      VECTOR2(NCOUNT,2) = YC(NRIGHT)-YC(NLEFT)
      VECTOR2(NCOUNT,3) = ZC(NRIGHT)-ZC(NLEFT)
      VEL(NCOUNT,2)= SQRT(VECTOR2(NCOUNT,1)*VECTOR2(NCOUNT,1)
     &    + VECTOR2(NCOUNT,2)*VECTOR2(NCOUNT,2)
     &       +VECTOR2(NCOUNT,3)*VECTOR2(NCOUNT,3))
C-----------------------------------------------------------
C    VEC2 IS SUM OF VEC1 AND VECTOR1
C-----------------------------------------------------------
      VECX2  = VECTOR1(NCOUNT,1)-VECX1
      VECY2  = VECTOR1(NCOUNT,2)-VECY1
      VECZ2  = VECTOR1(NCOUNT,3)-VECZ1
      VEC2   = SQRT(VECX2*VECX2+VECY2*VECY2+VECZ2*VECZ2)
C-----------------------------------------------------------
C    VECTOR1 AS UNIT VECTOR
C-----------------------------------------------------------
     VECTOR2(NCOUNT,1) = VECTOR2(NCOUNT,1)/VEL(NCOUNT,2)
     VECTOR2(NCOUNT,2) = VECTOR2(NCOUNT,2)/VEL(NCOUNT,2)
     VECTOR2(NCOUNT,3) = VECTOR2(NCOUNT,3)/VEL(NCOUNT,2)
     VEC2  = VEC1+VEC2
C-----------------------------------------------------------
C     QUADRATIC AX^2+BX+C  ARE CALCULATED
C-----------------------------------------------------------
      DELX1  =  VEC1
      DELX2  =  VEC2 - VEC1
      DELSQ1  =  VEC1*VEC1
      DELSQ2  = (VEC2*VEC2) - (VEC1*VEC1)
C   IF (NCOUNT .EQ. N*NT) THEN
C       WRITE (15,*) NCOUNT,B(NCOUNT)
C    ENDIF

      CONSTB = ((DELPHI2*DELSQ1)-(DELPHI1*DELSQ2))
     &    /((DELSQ1*DELX2)-(DELSQ2*DELX1))
      CONSTA = (DELPHI1-CONSTB*DELX1)/DELSQ1
     VEL(NCOUNT,2)=CONSTB+2.*CONSTA*VEC1
C------------------------------------------------------------
C
C------------------------------------------------------------
     CALL CONVERT(NCOUNT,VECTOR1)
     CALL CONVERT(NCOUNT,VECTOR2)
      CALL FINDBETA(VECTOR1,VECTOR2,BETA,NCOUNT)
      VEL(NCOUNT,1) = VEL(NCOUNT,1)
      VEL(NCOUNT,2) = VEL(NCOUNT,2)*SIN(BETA)
C-------------------------------------------------------
C   CALCULATING ABSOLUTE VALOCITY
C-------------------------------------------------------
C

SJOV=VEL(NCOUNT,1)*VEL(NCOUNT,1)+VEL(NCOUNT,2)*VEL(NCOUNT,2)
       VELABS(NCOUNT) = SQRT(SJOV)
601   CONTINUE
 801   CONTINUE

C-------------------------------------------------------
C   WRITING OUT THE VELOCITIES VS. ANGULAR POSITION
C-------------------------------------------------------
C
       KK=0
       DO 756 K=1,NX
        KK=((K-1)*NT)+1
 756    CONTINUE

C----------------------------------------------------------------
C
C----------------------------------------------------------------
       RETURN
       END

C----------------------------------------------------------------
C   END OF VELO
```

```
C-----------------------------------------------------------------

      SUBROUTINE CONVERT(N,VEC1)
        INCLUDE "PARAMETER.INC"
        COMMON /C1/ XP(MAXDIM,4),YP(MAXDIM,4),ZP(MAXDIM,4),PI,
     &         XC(MAXDIM),YC(MAXDIM),ZC(MAXDIM),V(3)
        COMMON /C2/ DC(3,3),ETA(5),QSI(5),X0,Y0,Z0,DA(MAXDIM),
     &         VNORMAL(MAXDIM,3),VELABS(MAXDIM),TRANS(MAXDIM,6)

      DIMENSION VEC1(MAXDIM,3)
      X = VEC1(N,1) - XP(N,1)
      Y = VEC1(N,2) - YP(N,1)
      Z = VEC1(N,3) - ZP(N,1)
C       X0 = TRANS(N,1)*(-XP(N,1))+TRANS(N,2)*(-YP(N,1))+
C    &       TRANS(N,3)*(-ZP(N,1))
C       Y0 = TRANS(N,4)*(-XP(N,1))+TRANS(N,5)*(-YP(N,1))+
C    &       TRANS(N,6)*(-ZP(N,1))
C       Z0 = VNORMAL(N,1)*(-XP(N,1))+VNORMAL(N,2)*(-YP(N,1))+
C    &       VNORMAL(N,3)*(-ZP(N,1))
C       VEC1(N,3)=0.0
      VEC1(N,1)=TRANS(N,1)*X+TRANS(N,2)*Y+TRANS(N,3)*Z
      VEC1(N,2)=TRANS(N,4)*X+TRANS(N,5)*Y+TRANS(N,6)*Z
      VEC1(N,3)=VNORMAL(N,1)*X+VNORMAL(N,2)*Y+VNORMAL(N,3)*Z
      Q=SQRT(VEC1(N,1)*VEC1(N,1)+VEC1(N,2)*VEC1(N,2)
     &     +VEC1(N,3)*VEC1(N,3))
      DO 2 I=1,3
        VEC1(N,I)=VEC1(N,I)/Q
 2      CONTINUE
       RETURN
      END


     SUBROUTINE  FINDBETA(V1,V2,BETA,NCOUNT)
        INCLUDE "PARAMETER.INC"
        DIMENSION V1(MAXDIM,3),V2(MAXDIM,3)
C---------------------------------------------------------
C   CALCULATES ANGLE BETA BETWEEN TWO VECTORS
C---------------------------------------------------------
     DOT = (V1(NCOUNT,1)*V2(NCOUNT,1))+
     &     (V1(NCOUNT,2)*V2(NCOUNT,2))+
     &     (V1(NCOUNT,3)*V2(NCOUNT,3))
     BETA=ACOS(DOT)
     RETURN
     END
```

Appendix E

```
      SUBROUTINE
INPUTVEHICLE(NPANEL,KODE,H,NX,NT,RHO,BLEN,QRM,CG,I7
     &   ,FNAME)
      INCLUDE "PARAMETER.INC"
      DATA COORTOL /0.0001/
      CHARACTER*14  TNAME
      CHARACTER*14 FNAME,GNAME,GEONAME, YOU*80
      COMMON /C1/ XP(MAXDIM,4),YP(MAXDIM,4),ZP(MAXDIM,4),PI,
     & XC(MAXDIM),YC(MAXDIM),ZC(MAXDIM),V(3)
      COMMON /C3/ BOUNDARY(MAXDIM),B(MAXDIM)
      COMMON /C4/ A(MAXDIM*MAXDIM)
      DIMENSION XBP(200),RBP(200),THET(200),
     & XPIN(200),RPIN(200),XIN(200),RIN(200),
     & QKSI(200),QLTB(200),RLBP(200),CT(200),CT2(200),QLTR(200),
     & NTE(200),YOU(80),ST(200),ST2(200),NPERB(200),CG(3),
     & AA2(200),BB(200),BB2(200),CONST(4),IP(4)

C-----------------------------------------------------
C
C    X IS FORWARD, Y IS TO STARBOARD AND Z IS DOWNWARD.
C    KN INPUT POINTS XPIN,RPIN FOR RADIUS VS X. POINT 1 IS TAIL.
C    NX CALCULATION AXIAL SPACES AND NT CALCULATION ANGULAR
SPACES.
C    NX1 CALCULATION POINTS FOR RADIUS VS X
C    V'S ARE 3 CPMTS. OF BODY VELY. BEING EQUIV. STREAM WHEN
NEGATIVE.
C    KODP=0 FOR NO VERT IMAGE, KODP=1 FOR PLANE H BELOW CL (PLANE
ABOVE
C    IS NEG H).  KODE=0 FOR NO L-R SYMMETRY, KODE=1 FOR L-R
SYMMETRY.
C    XG,YG,ZG IS THE POINT ABOUT WHICH MOMENTS ARE TAKEN.
C    NX=# PANEL COLUMNS
C-----------------------------------------------------
      I5=8
      I6=9
      I7=12
      I8=17
C     WRITE (*,'(A,$)')' TYPE INPUT FILE NAME  : '
C     READ (*,'(A)') FNAME
C-----------------------------------------------------
C    GETTING OUTPUT FILE NAME
C-----------------------------------------------------
C     WRITE (*,'(A,$)') ' TYPE THE OUTPUT FILE NAME  :'
```

```
C     READ (*,'(A)') TNAME
     TNAME = FNAME
        II=0
     DO 1 I=1,8
       IF( TNAME (I:I).NE.' ') THEN
         II=II+1
       ENDIF
 1     CONTINUE
C-----------------------------------------------------
C    MAKING OUTFILE NAME
C-----------------------------------------------------
       TNAME(II+1:II+1)='.'
       TNAME(II+2:II+2)='O'
       TNAME(II+3:II+3)='U'
       TNAME(II+4:II+4)='T'
       GNAME = TNAME
       GEONAME = TNAME
       GEONAME(II+2:II+2)='G'
       GEONAME(II+3:II+3)='E'
       GEONAME(II+4:II+4)='O'
C-----------------------------------------------------
     OPEN (I5,FILE = FNAME,STATUS = 'OLD')
     OPEN(I7,FILE=GNAME,STATUS='UNKNOWN')
     OPEN(I8,FILE=GEONAME,STATUS='UNKNOWN')
C-----------------------------------------------------
C    READING FILEHEADING 'YOU'
C-----------------------------------------------------
     READ (I5,11)(YOU(J),J=1,80)
C-----------------------------------------------------
C    NEW ADDITION THE DENSITY RHO IS NOW READ HERE
C    NOTE!!! FOR FOREIGNERS UNIT: SLUGS/AREA CUBED!!!
C-----------------------------------------------------
     READ (I5,*) KN,NT,NX,KODE,H,RHO,NPITCH,PITCH
     READ (I5,*) ICODE,EXECEN
     READ (I5,*) V(1),V(2),V(3)
     READ (I5,*) CG(1),CG(2),CG(3)
     WRITE (I8,11)(YOU(J),J=1,19)
     WRITE (I7,11)(YOU(J),J=1,19)
     WRITE (I7,*) 'KN,   NT,   NX,   KODE,   H,   RHO,
    &  NPITCH,   PITCH '
     WRITE (I7,15) KN,NT,NX,KODE,H,RHO,NPITCH,PITCH
     WRITE (I7,*) 'XVELOCITY,   YVELOCITY,   ZVELOCITY'
     WRITE (I7,13) V(1),V(2),V(3)
     WRITE (I7,*) 'MOMENT CALCULATED AROUND  (X,Y,Z):'
     WRITE (I7,13) CG(1),CG(2),CG(3)
```

```
C-------------------------------------------------------
C   R2 IS HALF THE DENSITY
C-------------------------------------------------------
      R2=RHO/2.
      H2=2.*H
C-------------------------------------------------------
C    QRM IS HALF DENSITY TIMES VELOCITY SQUARED
C    TIMES FRONTAL AREA
C-------------------------------------------------------
      DO 2 K=1,NX
        NTE(K)=NT
 2    CONTINUE
      NX1=NX+1
      NT1=NT+1
      BLEN=0.0
      QRM=0.0
C-------------------------------------------------------
      READ(I5,*)NCL,NCR,(XPIN(J),J=1,KN),(RPIN(J),J=1,KN),ESL,ESR
      QLEN=1.0
      BLEN=XPIN(KN)-XPIN(1)
      RMAX=0.0
      DO 7 J=1,KN
        IF (RPIN(J).GT.RMAX) RMAX=RPIN(J)
 7    CONTINUE
C-----------------------------------------------------------
C    QLEN IS THE DIMENTIONLESS LENGTH
C    XBP,RBP ARE THE CALC. POINTS .
C    AXIMUTHAL ANGLE SPACING IS THSTP.
C-----------------------------------------------------------
      QTHE=PI/NT
      RFAC=1.
      IF (ICODE.NE.0) THEN
        RFAC=SQRT(EXECEN)
        WRITE(*,*)RFAC
        H=H+(RMAX*(SQRT(EXECEN)-1))
        H2=2.*H
        WRITE(*,*)'NEW HEIGHT H : ',H
      ENDIF
C    RFAC=QTHE/(2.*SIN(QTHE/2.))
      DO 10 K=1,KN
        XIN(K)=XPIN(K)
 10     RIN(K)=RPIN(K)*RFAC
      RMAX=RMAX*RFAC
      XG=XG/QLEN
      YG=YG/QLEN
      ZG=ZG/QLEN
      H=H/QLEN
C-----------------------------------------------------------
C    EVALUATING THE SPLINE CUBICS FROM THE POINTS XIN, RIN GLYDK
C    RETURNS VALUES IN THE ARRAY AR
C-----------------------------------------------------------
      CALL UGLYDK(KN,NCL.NCR,XIN,RIN,ESL,ESR,AR,ICR)
      QLXN=XIN(KN)-XIN(1)
      QKSTP=PI/NX
C
      DO 20 K = 1, NX1
      QKSI(K) = PI - (K-1.)*QKSTP
      XBP(K) = XPIN(1) + 0.5*QLXN*(1.0 + COS(QKSI(K)))
20    CONTINUE
C
      THSTP=2.*PI/NT
C-----------------------------------------------------
C    2.*NT USED TO PUT ALL ELEMENTS ON STBD. AND L-R SYM TO BE
USED.
C-----------------------------------------------------
      THSPS=THSTP
C-----------------------------------------------------
C    RBP(K) IS BODY RADIUS AT K'TH LONGITUDINAL STATION
C    RLBP(K) IS RADIUS AT ELEMENT EDGE AT K'TH LONGITUDINAL
STATION
C    THET(K) IS AZIMUTH ANGLE AT K'TH PANEL BOUNDARY (TOWARDS +
ANGLE).
C-----------------------------------------------------
      CALL EVALDK(KN,NX1.XIN,XBP,RBP,AR,ICR)
C-----------------------------------------------------
C    PRINTING XVALUES AND RADIUS AT STATIONS
C-----------------------------------------------------
      RMAX2 =RMAX*RMAX
      IF ((ICODE.EQ.1).OR.(ICODE.EQ.2)) RMAX2 = RMAX2/EXECEN
      QRM=R2*(V(1)**2+V(2)**2+V(3)**2)*RMAX2*PI
      WRITE(*,*)'QRM :',QRM
      NPANEL=0
      WRITE (I7,*)' STATION #.    RADIUS,      XPOS'

        DO 57 III=1,NX+1
 57       WRITE (I7,45) III, RBP(III), XBP(III)

         IF(ICODE.EQ.0)THEN
           DO 30 KKK=1,NT1
             THET(KKK)=(KKK-1)*THSTP
```

```
          ST(KKK)=SIN(THET(KKK))
          ST2(KKK)=ST(KKK)*ST(KKK)
          CT(KKK)=COS(THET(KKK))
          CT2(KKK)=CT(KKK)*CT(KKK)
30       CONTINUE
      ELSE
        CALL ELLIPSE(EXECEN,NT1,THET)
        DO 60 KKK=1,NT1
          ST(KKK)=SIN(THET(KKK))
          ST2(KKK)=ST(KKK)*ST(KKK)
          CT(KKK)=COS(THET(KKK))
          CT2(KKK)=CT(KKK)*CT(KKK)
60       CONTINUE
      ENDIF

    DO 50 K=1,NX
    IF(ICODE.EQ.1)THEN
        DO 55 J=1,NX
          BB(J)=RBP(J)/EXECEN
          BB2(J)=BB(J)*BB(J)
          AA2(J)=RBP(J)*RBP(J)
55       CONTINUE
      ENDIF

    IF(ICODE.EQ.2)THEN
        DO 56 J=1,NX
          BB(J)=RBP(J)/EXECEN
          AA2(J)=BB(J)*BB(J)
          BB2(J)=RBP(J)*RBP(J)
56       CONTINUE
      ENDIF

      DO 50 L=1,NT
        LL=L+1
        KK=(K+1)
        NPANEL = NPANEL+1
C----------------------------------------------------
C    PANEL COORDINATES CALCULATED
C    X(NPANEL,1-4)
C----------------------------------------------------
C
C----------------------------------------------------
C    IN CASE OF AXISYMMETRIC BODY
C----------------------------------------------------
       IF(ICODE.EQ.0)THEN
       XP(NPANEL,1)=XBP(K)
       YP(NPANEL,1)=RBP(K)*ST(L)
       ZP(NPANEL,1)=RBP(K)*CT(L)
       XP(NPANEL,2)=XBP(K)
       YP(NPANEL,2)=RBP(K)*ST(LL)
       ZP(NPANEL,2)=RBP(K)*CT(LL)
       XP(NPANEL,3)=XBP(KK)
       YP(NPANEL,3)=RBP(KK)*ST(LL)
       ZP(NPANEL,3)=RBP(KK)*CT(LL)
       XP(NPANEL,4)=XBP(KK)
       YP(NPANEL,4)=RBP(KK)*ST(L)
       ZP(NPANEL,4)=RBP(KK)*CT(L)

       XDIF12 = ABS(XP(NPANEL,1)-XP(NPANEL,2))
       YDIF12 = ABS(YP(NPANEL,1)-YP(NPANEL.2))
       ZDIF12 = ABS(ZP(NPANEL,1)-ZP(NPANEL.2))
       XDIF43 = ABS(XP(NPANEL,4)-XP(NPANEL.3))
       YDIF43 = ABS(YP(NPANEL,4)-YP(NPANEL,3))
       ZDIF43 = ABS(ZP(NPANEL,4)-ZP(NPANEL.3))

      ENDIF
C-----------------------------------------------------
C   IN CASE OF ELLIPTICAL CROSS SECTION
C-----------------------------------------------------

      IF ((ICODE.EQ.1).OR.(ICODE.EQ.2)) THEN

C-----------------------------------------------------
C   NOT FIRST OR LAST BAND
C-----------------------------------------------------
       IF ((K.NE.1).AND.(K.NE.NX)) THEN
         CONST(1)=SQRT((AA2(K)*BB2(K))/((AA2(K)*ST2(L))+
     &       (BB2(K)*CT2(L))))
         CONST(2)=SQRT((AA2(K)*BB2(K))/((AA2(K)*ST2(LL))+
     &       (BB2(K)*CT2(LL))))
         CONST(3)=SQRT((AA2(KK)*BB2(KK))/((AA2(KK)*ST2(LL))+
     &       (BB2(KK)*CT2(LL))))
        CONST(4)=SQRT((AA2(KK)*BB2(KK))/((AA2(KK)*ST2(L))+
     &       (BB2(KK)*CT2(L))))
      ENDIF


C-----------------------------------------------------
C   FIRST BAND
C-----------------------------------------------------
```

```
   IF (K.EQ.1) THEN
     CONST(1) = 0.0
     CONST(2) = 0.0
     CONST(3) = SQRT((AA2(KK)*BB2(KK))/((AA2(KK)*ST2(LL))+
  &              (BB2(KK)*CT2(LL))))
     CONST(4) = SQRT((AA2(KK)*BB2(KK))/((AA2(KK)*ST2(L))+
  &              (BB2(KK)*CT2(L))))
   ENDIF


C-----------------------------------------------------
C    LAST BAND
C-----------------------------------------------------
   IF (K.EQ.NX) THEN
     CONST(1) = SQRT((AA2(K)*BB2(K))/((AA2(K)*ST2(L))+
  &              (BB2(K)*CT2(L))))
     CONST(2) = SQRT((AA2(K)*BB2(K))/((AA2(K)*ST2(LL))+
  &              (BB2(K)*CT2(LL))))
     CONST(3) = 0.0
     CONST(4) = 0.0
   ENDIF

       XP(NPANEL,1) = XBP(K)
       YP(NPANEL,1) = CONST(1)*ST(L)
       ZP(NPANEL,1) = CONST(1)*CT(L)
       XP(NPANEL,2) = XBP(K)
       YP(NPANEL,2) = CONST(2)*ST(LL)
       ZP(NPANEL,2) = CONST(2)*CT(LL)
       XP(NPANEL,3) = XBP(KK)
       YP(NPANEL,3) = CONST(3)*ST(LL)
       ZP(NPANEL,3) = CONST(3)*CT(LL)
       XP(NPANEL,4) = XBP(KK)
       YP(NPANEL,4) = CONST(4)*ST(L)
       ZP(NPANEL,4) = CONST(4)*CT(L)

       XDIF12 = ABS(XP(NPANEL,1)-XP(NPANEL,2))
       YDIF12 = ABS(YP(NPANEL,1)-YP(NPANEL.2))
       ZDIF12 = ABS(ZP(NPANEL,1)-ZP(NPANEL,2))
       XDIF43 = ABS(XP(NPANEL,4)-XP(NPANEL,3))
       YDIF43 = ABS(YP(NPANEL,4)-YP(NPANEL,3))
       ZDIF43 = ABS(ZP(NPANEL,4)-ZP(NPANEL,3))

     ENDIF
```

```
       IF((XDIF12.LT.COORTOL).AND.(YDIF12.LT.COORTOL)
     &  .AND.(ZDIF12.LT.COORTOL)) THEN
         XP(NPANEL,2)=(XP(NPANEL,1) +XP(NPANEL,3))/2
         YP(NPANEL,2)=(YP(NPANEL,1) +YP(NPANEL,3))/2
         ZP(NPANEL,2)=(ZP(NPANEL,1) +ZP(NPANEL,3))/2
      ENDIF

      IF((XDIF43.LT.COORTOL).AND.(YDIF43.LT.COORTOL)
     &   .AND.(ZDIF43.LT.COORTOL)) THEN
        XP(NPANEL,3)=(XP(NPANEL,2) +XP(NPANEL,4))/2
        YP(NPANEL,3)=(YP(NPANEL,2) +YP(NPANEL,4))/2
        ZP(NPANEL,3)=(ZP(NPANEL,2) +ZP(NPANEL,4))/2
      ENDIF

 50   CONTINUE
      IF((KODE.NE.0).AND.(RMAX.GE.H))THEN
       WRITE(*,*)
       WRITE(*,*)'        ++++++++++++++++++++++++
     &+++++++++++++++++'
       WRITE(*,*)'        +
     &            +'
       WRITE(*,*)'        + THE VEHICLE IS PENETR
     &ATING THE WALL  +'
       WRITE(*,*)'        +     TRY A GREATER
     & VALUE FOR H     +'
       WRITE(*,*)'        +
     &            +'
       WRITE(*,*)'        ++++++++++++++++++++++++
     &+++++++++++++++++'

       WRITE(*,*)
       STOP
      ENDIF
C----------------------------------------------------------
C   CALCULATING BODY WITH PITCH
C----------------------------------------------------------
      ZMIN=0.0
      IF (NPITCH.EQ.0) THEN
        IF(ICODE.EQ.0)ZMIN=-RMAX
        IF(ICODE.EQ.1)ZMIN=-RMAX
        IF(ICODE.EQ.2)ZMIN=-RMAX/EXECEN
      ENDIF
      IF (NPITCH.NE.0) THEN
        WRITE (I7,*)'PITCH :',PITCH
        PITCH=PITCH*PI/180
      WRITE (*,*)'PITCH :',PITCH
      SINP=SIN(PITCH)
      COSP=COS(PITCH)
      DO 200 K=1,NPANEL
        DO 220 J=1,4
          XP1=CG(1)+((XP(K,J)-CG(1))*COSP)-((ZP(K,J)-CG(3))*SINP)
          ZP1=CG(3)+((ZP(K,J)-CG(3))*COSP)+((XP(K,J)-CG(1))*SINP)
          IF(ZP1.LT.ZMIN)ZMIN=ZP1
          XP(K,J)=XP1
          ZP(K,J)=ZP1
 220      CONTINUE
 200    CONTINUE
C
    ENDIF
    IF ((ABS(ZMIN).GE.H).AND.(KODE.NE.0).AND.(NPITCH.NE.0))THEN
      WRITE (*,*)' THE BODY HAS PENETRATED THE WALL'
      WRITE (*,*)' PLEASE TRY AGAIN WITH GREATER GAP'
      WRITE (*,*)' OR SMALLER PITCH ANGLE'
      STOP
    ENDIF
    WRITE (I7,231)ZMIN+H
C-----------------------------------------------------------
C   WRITING TO .GEO FILE
C-----------------------------------------------------------
     WRITE(I8,*)'VARIABLES = X,Y,Z'
     WRITE(I8,300)4*NPANEL,NPANEL
     DO 25 I=1,NPANEL
       DO 23 KK=1,4
         WRITE (I8,109)XP(I,KK),YP(I,KK),ZP(I,KK)
 23      CONTINUE
 25    CONTINUE
     DO 210 I=1,4*NPANEL,4
       IP(1)=I
       IP(2)=I+1
       IP(3)=I+2
       IP(4) = I+3
       WRITE(I8,320)IP(1),IP(2),IP(3),IP(4)
 210   CONTINUE

     WRITE(*,*)NPANEL
C-----------------------------------------------------------
C   FORMAT LISTING
C-----------------------------------------------------------

 11   FORMAT (80A4)
 13   FORMAT(1X,3F12.3)
 14   FORMAT(1X,4I5,2F7.3)
 15   FORMAT(1X,4I5,2F8.3,I5,F8.3)
 45   FORMAT(1X,I8,3X,4F12.4)
 107  FORMAT(1X,I5)
 109  FORMAT(1X,3F12.5)
 231  FORMAT ('MINIMUM GAP :',F8.4)
 300  FORMAT('ZONE T="1", I=',I4,', J=',I4,', F=FEPOINT')
 310  FORMAT(A)
 312  FORMAT('TITLE="',A,'"')
 320  FORMAT(4I6)
C-----------------------------------------------------------
C    FORMAT LISTING  END
C-----------------------------------------------------------
     RETURN
     END
```

Appendix F

```
      SUBROUTINE CENTER(NUMBER)
      include "parameter.inc"
C-------------------------------------------------------------------
C     This subroutine calculates the centroid of each panel
C     in the global coordinate system
C-------------------------------------------------------------------
      COMMON /C1/ XP(MAXDIM,4),YP(MAXDIM,4),ZP(MAXDIM,4),PI,
     &            XC(MAXDIM),YC(MAXDIM),ZC(MAXDIM),V(3)
      COMMON /C2/ DC(3,3),ETA(5),QSI(5),X0,Y0,Z0,DA(MAXDIM),
     & vnormal(MAXDIM,3),velabs(maxdim),trans(maxdim,6)
      COMMON /C3/ BOUNDARY(MAXDIM),B(MAXDIM)
      DIMENSION D(4),ES(4),QS(4),XL(4),YL(4),ZL(4),XQ(4),YQ(4),ZQ(4)
C-------------------------------------------------------------------
C     Assign the corner coordinates of each panel represented
C     by its global mesh number
C-------------------------------------------------------------------
      DO 40 I=1,NUMBER
      DO 10 K=1,4
      XL(K)=XP(I,K)
      YL(K)=YP(I,K)
      ZL(K)=ZP(I,K)
10    CONTINUE
C-------------------------------------------------------------------
C     Find the average of panel corner coordinates
C-------------------------------------------------------------------
      XAV=0.25*(XL(1)+XL(2)+XL(3)+XL(4))
      YAV=0.25*(YL(1)+YL(2)+YL(3)+YL(4))
      ZAV=0.25*(ZL(1)+ZL(2)+ZL(3)+ZL(4))
C-------------------------------------------------------------------
C     Number the corner points of each panel in a clock-wise manner
C     when viewed from above, then find a normal vector to two vectors
C     that connect the pair of points (1,3) and (2,4)
C-------------------------------------------------------------------
      QNX=(YL(4)-YL(2))*(ZL(3)-ZL(1))-(ZL(4)-ZL(2))*(YL(3)-YL(1))
      QNY=(ZL(4)-ZL(2))*(XL(3)-XL(1))-(ZL(3)-ZL(1))*(XL(4)-XL(2))
      QNZ=(XL(4)-XL(2))*(YL(3)-YL(1))-(YL(4)-YL(2))*(XL(3)-XL(1))
      QMAG=1./SQRT(QNX*QNX+QNY*QNY+QNZ*QNZ)
C-------------------------------------------------------------------
C     Unit normal vector
C-------------------------------------------------------------------
      DC(3,1)=QNX*QMAG
      DC(3,2)=QNY*QMAG
      DC(3,3)=QNZ*QMAG
      vnormal(I,1) = DC(3,1)
      vnormal(I,2) = DC(3,2)
      vnormal(I,3) = DC(3,3)
 13   format(1x,i5,3f12.6)
C-----------------------------------------------------------------
C     Define a plane with this normal vector and the average point
C     Find the new corner coordinates of the panel by projecting the
C     original corner coordinates onto that plane as these original
C     coordinates in space may not necessarily form a flat plane
C     Corner coordinates in the global coordinate system
C-----------------------------------------------------------------
      DO 20 K=1,4
      D(K)=DC(3,1)*(XAV-XL(K))+DC(3,2)*(YAV-YL(K))+DC(3,3)*(ZAV-ZL(K))
      XQ(K)=XL(K)+DC(3,1)*D(K)
      YQ(K)=YL(K)+DC(3,2)*D(K)
      ZQ(K)=ZL(K)+DC(3,3)*D(K)
 20   CONTINUE
C-----------------------------------------------------------------
C     Elements of the transformation matrix between the global and
C     local coordinate systems
C-----------------------------------------------------------------
      QMAG=1./SQRT((XL(3)-XL(1))**2+(YL(3)-YL(1))**2+(ZL(3)-ZL(1))**2)
      DC(1,1)=(XL(3)-XL(1))*QMAG
      DC(1,2)=(YL(3)-YL(1))*QMAG
      DC(1,3)=(ZL(3)-ZL(1))*QMAG
      DC(2,1)=DC(3,2)*DC(1,3)-DC(3,3)*DC(1,2)
      DC(2,2)=DC(3,3)*DC(1,1)-DC(3,1)*DC(1,3)
      DC(2,3)=DC(3,1)*DC(1,2)-DC(3,2)*DC(1,1)
      trans(i,1)=dc(1,1)
      trans(i,2)=dc(1,2)
      trans(i,3)=dc(1,3)
      trans(i,4)=dc(2,1)
      trans(i,5)=dc(2,2)
      trans(i,6)=dc(2,3)
C-----------------------------------------------------------------
C     Corner coordinates in the global coordinate system based on the
C     average point as origin
C-----------------------------------------------------------------
      DO 30 K=1,4
      QS(K)=DC(1,1)*(XQ(K)-XAV)+DC(1,2)*(YQ(K)-YAV)+DC(1,3)*(ZQ(K)-ZAV)
      ES(K)=DC(2,1)*(XQ(K)-XAV)+DC(2,2)*(YQ(K)-YAV)+DC(2,3)*(ZQ(K)-ZAV)
 30   CONTINUE
C-----------------------------------------------------------------
C     The centroid in the local coordinate system with the average
C     point as origin
```

```
C------------------------------------------------------------------
      QS0=(QS(4)*(ES(1)-ES(2))+QS(2)*(ES(4)-ES(1)))/(3.*(ES(2)-ES(4)))
      ES0=-ES(1)/3.
C------------------------------------------------------------------
C     Panel centroids in the global coordinate system
C------------------------------------------------------------------
      XC(I)=XAV+DC(1,1)*QS0+DC(2,1)*ES0
      YC(I)=YAV+DC(1,2)*QS0+DC(2,2)*ES0
      ZC(I)=ZAV+DC(1,3)*QS0+DC(2,3)*ES0
C------------------------------------------------------------------
C     the boundary condition velocity dot normal vector
C------------------------------------------------------------------
      BOUNDARY(i) = V(1)* DC(3,1)+V(2)* DC(3,2)+V(3)* DC(3,3)
40    CONTINUE
C------------------------------------------------------------------
C     Exit   CENTER()
C------------------------------------------------------------------
      RETURN
      END
C
C
C
C
      SUBROUTINE COFMAT(NTOT,kode,h)
C------------------------------------------------------------------
C     This subroutine evaluates the influence coefficients matrix
C     Boundary integral equation is formed by a mixed distribution
C     of sources and normal dipoles (Green's formulation)
C     Source and dipole effects are computed by the exact formulation
C     which follows from Newman, J. Eng. Math., 1986
C------------------------------------------------------------------
      include "parameter.inc"
C
      COMMON /C1/ XP(MAXDIM,4),YP(MAXDIM,4),ZP(MAXDIM,4),PI,
     &            XC(MAXDIM),YC(MAXDIM),ZC(MAXDIM),V(3)
      COMMON /C2/ DC(3,3),ETA(5),QSI(5),X0,Y0,Z0,DA(MAXDIM),
     & vnormal(maxdim,3),velabs(maxdim),trans(maxdim,6)
      COMMON /C3/ BOUNDARY(MAXDIM),B(MAXDIM)
      COMMON /C4/ A(MAXDIM*MAXDIM)
      DIMENSION GARR(MAXDIM),HARR(MAXDIM)
C------------------------------------------------------------------
C     Tolerance in distance between the source panel plane and the
C     field point (field point assumed to be on the source panel plane
C     whenever this distance is smaller than ZTOL)
C------------------------------------------------------------------
```

```
      DATA ZTOL /0.000001/
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     Initialize the r.h.s. vector
C-----------------------------------------------------------------------
      DO 10 I=1,NTOT
      B(I)=0.0
 10   CONTINUE
C-----------------------------------------------------------------------
C     Initialize the influence matrix
C-----------------------------------------------------------------------
      DO 20 I=1,NTOT*NTOT
      A(I)=0.0
 20   continue
C-----------------------------------------------------------------------
C     Start with a source panel, and find the influence due to source
C     and dipole parts over each field point which is the centroid of
C     other panels on boundaries of the domain
C-----------------------------------------------------------------------
      TPI=2.*PI
C-----------------------------------------------------------------------
C     Start with source points on the free surface
C-----------------------------------------------------------------------
      area=0.0
      nkode=1
      if(kode.ne.0) nkode=2
      do 35 k=1,nkode
C
      DO 40 ISP=1,NTOT
C-----------------------------------------------------------------------
C     Form the quadrilateral panel plane and find the projected corner
C     coordinates in the local coordinate system
C-----------------------------------------------------------------------
C
      CALL QPLANE(ISP,h,k)
      QDIF1=QSI(2)-QSI(1)
      QDIF2=QSI(3)-QSI(2)
      QDIF3=QSI(4)-QSI(3)
      QDIF4=QSI(1)-QSI(4)
      EDIF1=ETA(2)-ETA(1)
      EDIF2=ETA(3)-ETA(2)
      EDIF3=ETA(4)-ETA(3)
      EDIF4=ETA(1)-ETA(4)
      S1=SQRT(QDIF1*QDIF1+EDIF1*EDIF1)
      S2=SQRT(QDIF2*QDIF2+EDIF2*EDIF2)
      S3=SQRT(QDIF3*QDIF3+EDIF3*EDIF3)
      S4=SQRT(QDIF4*QDIF4+EDIF4*EDIF4)
C-----------------------------------------------------------------
C     Find the panel area of the source point
C-----------------------------------------------------------------
      DA(ISP)=0.5*((QSI(3)-QSI(1))*(ETA(2)-ETA(4)))
      area=area+da(isp)
C-----------------------------------------------------------------
C     Start the loop for field points over the entire domain
C-----------------------------------------------------------------
      DO 30 IFP=1,NTOT
      XR=DC(1,1)*(XC(IFP)-X0)+DC(1,2)*(YC(IFP)-Y0)+DC(1,3)*(ZC(IFP)-Z0)
      YR=DC(2,1)*(XC(IFP)-X0)+DC(2,2)*(YC(IFP)-Y0)+DC(2,3)*(ZC(IFP)-Z0)
      ZR=DC(3,1)*(XC(IFP)-X0)+DC(3,2)*(YC(IFP)-Y0)+DC(3,3)*(ZC(IFP)-Z0)
      R1=SQRT((XR-QSI(1))*(XR-QSI(1))+(YR-ETA(1))*(YR-ETA(1))+ZR*ZR)
      R2=SQRT((XR-QSI(2))*(XR-QSI(2))+(YR-ETA(2))*(YR-ETA(2))+ZR*ZR)
      R3=SQRT((XR-QSI(3))*(XR-QSI(3))+(YR-ETA(3))*(YR-ETA(3))+ZR*ZR)
      R4=SQRT((XR-QSI(4))*(XR-QSI(4))+(YR-ETA(4))*(YR-ETA(4))+ZR*ZR)
C-----------------------------------------------------------------
C     Self induced velocity when the field point approaches the
C     source point (field point = source point)
C-----------------------------------------------------------------
      IF(IFP.EQ.ISP .and. k.eq.1) THEN
      ZR=0.0
      HARR(IFP)=-TPI
C-----------------------------------------------------------------
C     Field point is not on the source plane if the following
C     holds true
C-----------------------------------------------------------------
      ELSE
      IF(ABS(ZR).GT.ZTOL) THEN
      S11=(EDIF1*((XR-QSI(1))*(XR-QSI(1))+ZR*ZR)-QDIF1*
     &    (XR-QSI(1))*(YR-ETA(1)))
      S12=(EDIF2*((XR-QSI(2))*(XR-QSI(2))+ZR*ZR)-QDIF2*
     &    (XR-QSI(2))*(YR-ETA(2)))
      S13=(EDIF3*((XR-QSI(3))*(XR-QSI(3))+ZR*ZR)-QDIF3*
     &    (XR-QSI(3))*(YR-ETA(3)))
      S14=(EDIF4*((XR-QSI(4))*(XR-QSI(4))+ZR*ZR)-QDIF4*
     &    (XR-QSI(4))*(YR-ETA(4)))
      S21=(EDIF1*((XR-QSI(2))*(XR-QSI(2))+ZR*ZR)-QDIF1*
     &    (XR-QSI(2))*(YR-ETA(2)))
      S22=(EDIF2*((XR-QSI(3))*(XR-QSI(3))+ZR*ZR)-QDIF2*
     &    (XR-QSI(3))*(YR-ETA(3)))
      S23=(EDIF3*((XR-QSI(4))*(XR-QSI(4))+ZR*ZR)-QDIF3*
     &    (XR-QSI(4))*(YR-ETA(4)))
```

```
      S24=(EDIF4*((XR-QSI(1))*(XR-QSI(1))+ZR*ZR)-QDIF4*
     &    (XR-QSI(1))*(YR-ETA(1)))
      C11=R1*QDIF1*ZR
      C12=R2*QDIF2*ZR
      C13=R3*QDIF3*ZR
      C14=R4*QDIF4*ZR
      C21=R2*QDIF1*ZR
      C22=R3*QDIF2*ZR
      C23=R4*QDIF3*ZR
      C24=R1*QDIF4*ZR
C------------------------------------------------------------------
C   Compute the exact dipole effect due to constant strength
C   Equation-2.14 in Newman, 1986
C------------------------------------------------------------------
      HARR(IFP)=ATAN2((S11*C21-S21*C11),(C11*C21+S11*S21))+
     &        ATAN2((S12*C22-S22*C12),(C12*C22+S12*S22))+
     &        ATAN2((S13*C23-S23*C13),(C13*C23+S13*S23))+
     &        ATAN2((S14*C24-S24*C14),(C14*C24+S14*S24))
C------------------------------------------------------------------
C   Zero induced velocity when the field point is on the source plane
C------------------------------------------------------------------
      ELSE
      ZR=0.0
      HARR(IFP)=0.0
      ENDIF
      ENDIF
C------------------------------------------------------------------
C   Compute the exact source effect due to constant strength
C   Equation-3.10 in Newman, 1986
C------------------------------------------------------------------
      ARG1=(R1+R2+S1)/(R1+R2-S1)
      ARG2=(R2+R3+S2)/(R2+R3-S2)
      ARG3=(R3+R4+S3)/(R3+R4-S3)
      ARG4=(R4+R1+S4)/(R4+R1-S4)
      G1=((XR-QSI(1))*EDIF1-(YR-ETA(1))*QDIF1)*ALOG(ARG1)/S1
      G2=((XR-QSI(2))*EDIF2-(YR-ETA(2))*QDIF2)*ALOG(ARG2)/S2
      G3=((XR-QSI(3))*EDIF3-(YR-ETA(3))*QDIF3)*ALOG(ARG3)/S3
      G4=((XR-QSI(4))*EDIF4-(YR-ETA(4))*QDIF4)*ALOG(ARG4)/S4
      GARR(IFP)=G1+G2+G3+G4-ZR*HARR(IFP)
C------------------------------------------------------------------
C   Store the r.h.s and the influence coefficient array
C   The BOUNDARY array contains the dot product of the stream velocity
C   and the normal vector of the N'th element. Remember normal vector
C   goes out of body.
C------------------------------------------------------------------
```

```
      B(IsP)=b(ISP)+GARR(IFP)*BOUNDARY(IFP)
      A((ISP-1)*NTOT+IFP)=A((ISP-1)*NTOT+IFP)+HARR(IFP)
 30   CONTINUE
 40   CONTINUE
 35   continue
C-----------------------------------------------------------------
C     EXIT  COFMAT
C-----------------------------------------------------------------
      RETURN
      END

C
      SUBROUTINE QPLANE(ISP,h,ntest)
      include "parameter.inc"
C-----------------------------------------------------------------
C     This subroutine forms a quadrilateral source plane and
C     determines the corner coordinates in the local coordinate
C     system on the projected plane
C-----------------------------------------------------------------
      COMMON /C1/ X(MAXDIM,4),Y(MAXDIM,4),Z(MAXDIM,4),PI,
     &            XC(MAXDIM),YC(MAXDIM),ZC(MAXDIM),V(3)
      COMMON /C2/ DC(3,3),ETA(5),QSI(5),X0,Y0,Z0,DA(MAXDIM),
     & vnormal(maxdim,3),velabs(maxdim,3),trans(maxdim,6)
      DIMENSION D(4),XL(4),YL(4),ZL(4),XQ(4),YQ(4),ZQ(4)
C-----------------------------------------------------------------
C     Variables used here are similar to those in CENTER
C-----------------------------------------------------------------
      if(ntest.eq.1) then
      DO 10 K=1,4
        XL(K)=X(ISP,K)
        YL(K)=Y(ISP,K)
        ZL(K)=Z(ISP,K)
 10   CONTINUE
      else
      do 15 k=1,4
        l=k
        if(k.eq.2) l=4
        if(k.eq.4) l=2
        xl(l)=x(isp,k)
        yl(l)=y(isp,k)
        zl(l)=-2.0*h-z(isp,k)
15    continue
      endif
C-----------------------------------------------------------------
C     Local axis 'ksi' goes through corner points <1> and <3>
```

```
C     Note that corner points are numbered in a clock-wise
C     manner when the panel is viewed from above and this
C     convention is maintained throughout the program
C-----------------------------------------------------------------
      QNX=(YL(4)-YL(2))*(ZL(3)-ZL(1))-(ZL(4)-ZL(2))*(YL(3)-YL(1))
      QNY=(ZL(4)-ZL(2))*(XL(3)-XL(1))-(ZL(3)-ZL(1))*(XL(4)-XL(2))
      QNZ=(XL(4)-XL(2))*(YL(3)-YL(1))-(YL(4)-YL(2))*(XL(3)-XL(1))
      QMAG=1./SQRT(QNX*QNX+QNY*QNY+QNZ*QNZ)
C-----------------------------------------------------------------
C     unit normal vector to the plane in global coordinates
C-----------------------------------------------------------------
      DC(3,1)=QNX*QMAG
      DC(3,2)=QNY*QMAG
      DC(3,3)=QNZ*QMAG
C-----------------------------------------------------------------
C     the avarage point (xav,yav,zav)
C-----------------------------------------------------------------
      XAV=0.25*(XL(1)+XL(2)+XL(3)+XL(4))
      YAV=0.25*(YL(1)+YL(2)+YL(3)+YL(4))
      ZAV=0.25*(ZL(1)+ZL(2)+ZL(3)+ZL(4))
C-----------------------------------------------------------------
C     corner coordinates projected into the plane of the element
c     in global coordinates
C-----------------------------------------------------------------
      DO 20 K=1,4
      D(K)=DC(3,1)*(XAV-XL(K))+DC(3,2)*(YAV-YL(K))+DC(3,3)*(ZAV-ZL(K))
      XQ(K)=XL(K)+DC(3,1)*D(K)
      YQ(K)=YL(K)+DC(3,2)*D(K)
      ZQ(K)=ZL(K)+DC(3,3)*D(K)
 20   continue
C-----------------------------------------------------------------
C     length of vector t1 qmag, t1 goes from point #1 to point #3
C-----------------------------------------------------------------
      QMAG=1./SQRT((XL(3)-XL(1))**2+(YL(3)-YL(1))**2+(ZL(3)-ZL(1))**2)
      DC(1,1)=(XL(3)-XL(1))*QMAG
      DC(1,2)=(YL(3)-YL(1))*QMAG
      DC(1,3)=(ZL(3)-ZL(1))*QMAG
C-----------------------------------------------------------------
C     vector t2 is calculated
C-----------------------------------------------------------------
      DC(2,1)=DC(3,2)*DC(1,3)-DC(3,3)*DC(1,2)
      DC(2,2)=DC(3,3)*DC(1,1)-DC(3,1)*DC(1,3)
      DC(2,3)=DC(3,1)*DC(1,2)-DC(3,2)*DC(1.1)
C-----------------------------------------------------------------
C     Global coordinates of the origin of the local coordinate system
```

```
C----------------------------------------------------------------------
      X0=XQ(1)
      Y0=YQ(1)
      Z0=ZQ(1)
C----------------------------------------------------------------------
C   Corner coordinates of the quadrilateral plane in the local
C   coordinate system, EQUATION (80) p 72 Hess and Smith 1962
C----------------------------------------------------------------------
      DO 30 K=1,4
      QSI(K)=DC(1,1)*(XQ(K)-X0)+DC(1,2)*(YQ(K)-Y0)+DC(1,3)*(ZQ(K)-Z0)
      ETA(K)=DC(2,1)*(XQ(K)-X0)+DC(2,2)*(YQ(K)-Y0)+DC(2,3)*(ZQ(K)-Z0)
 30   CONTINUE
C----------------------------------------------------------------------
C   Exit  qplane
C----------------------------------------------------------------------
      RETURN
      END
```

Appendix G

```
      SUBROUTINE SOLVER(SIG,RHSM,TOL,NORD,NBLOCK,NPERB,
     &                    NPB,IPASS,NFOUT)
C--------------------------------------------------------------------
C    BLOCK ITERATIVE SOLUTION OF COLUMN STORED MATRIX
C
C    SIG    : Solution vector (output)
C    RHSM   : Right hand side vector (input)
C    TOL    : Maximum residual for convergence (input)
C    NORD   : Order of matrix (input)
C    NBLOCK : Number of diagonal blocks in matrix (input)
C    NPERB  : Array of number of equations in
C             each diagonal block (input)
C    NPB    : Number of equations in each diagonal block (input)
C    ITMAX  : Maximum number of iterations allowed (input)
C    IPASS  : Pass counter for rhs (input)
C           = 1 first rhs
C           = 2 second rhs, etc.
C    NFOUT  : Unit number for printout of convergence (input)
C--------------------------------------------------------------------
C    UABU  :  Storage array for upper triag blocks
C    UABL  :  Storage array for lower triag blocks
C    UABD  :  Storage array for diagonal blocks
C
C    Arrays URS1 and URS2 are used for accelerator files
C    and are swapped after each iteration
C
C    DSIG  : Solution perturbation vector
C    RES   : Current residual vector
C
C
C    Acknowledgment :
C    ---------------
C    This routine (SOLVER) has been provided by Dr. D.Greeley of
C    Atlantic Research, Inc., and modified to accommodate all of
C    the I/O operations in memory
C
C--------------------------------------------------------------------
      INCLUDE 'parameter.inc'
C--------------------------------------------------------------------
C    NEQ, NBLK and NNB are assigned constant values to be able
C    to share the storage in array A which is in no longer use
C    These three quantities must equal NTOT, NBLOCK and NPB
```

```
C-------------------------------------------------------------------
      PARAMETER(NEQ=900,NBLK=30,NNB=30,ITMAX=50)
      DIMENSION RHSM(1),SIG(1),NPERB(1)
c      DIMENSION ATEM(NEQ),DSIG(NEQ),RES(NEQ),RESNEW(NEQ),
c     &        SIGNEW(NEQ),AD(NNB*NNB),F(NNB),IPVT(NNB),
c     &        P(NNB.NNB),Q11(NNB,NNB),Q12(NNB),RHSD(NNB),
c     &        WORK(NNB),UABD(NEQ*(NNB+1)),
c     &        UABL(NNB*NNB*NBLK*(NBLK-1)/2),
c     &        UABU(NNB*NNB*NBLK*(NBLK-1)/2),
c     &        URS1(2*NEQ*NNB),URS2(2*NEQ*NNB)
      DIMENSION ATEM(NEQ),DSIG(NEQ),RES(NEQ),RESNEW(NEQ),
     &        SIGNEW(NEQ),AD(ITMAX*ITMAX),F(ITMAX),IPVT(ITMAX),
     &        P(ITMAX.ITMAX),Q11(ITMAX,ITMAX),Q12(ITMAX),RHSD(ITMAX),
     &        WORK(ITMAX),UABD(NEQ*(NNB+1)),
     &        UABL(NNB*NNB*NBLK*(NBLK-1)/2),
     &        UABU(NNB*NNB*NBLK*(NBLK-1)/2),
     &        URS1(2*NEQ*NNB),URS2(2*NEQ*NNB)
      EQUIVALENCE (AD(1),Q11(1,1)),(F(1),RHSD(1))
C-------------------------------------------------------------------
C                   *** WARNING ***
C     Replace the dimension of AD,F,IPVT,Q11,Q12,P,RHSD and WORK by
C     'ITMAX' if : ITMAX > NPB
C-------------------------------------------------------------------
C     Share the memory storage of vector A in common block C3
C     as it is in no longer use (written as a binary file)
C-------------------------------------------------------------------
      COMMON /C4/ A(maxdim*maxdim)
C-------------------------------------------------------------------
C     Open the convergence information file
C-------------------------------------------------------------------
      OPEN(NFOUT,FILE='mat.inf',STATUS='UNKNOWN')
      MAXDM=MAX0(NPB,ITMAX)
      NPERMX=NPB
      IF(IPASS.GT.1) GO TO 500
C-------------------------------------------------------------------
C     Separate matrix into lower triangular, diagonal,
C     and upper triangular blocks, and factor diagonal blocks
C-------------------------------------------------------------------
      I1=0
      I2=0
      I3=0
      IDSTRT=1
      NCOUNT=0
      DO 490 NB=1,NBLOCK
      NPER=NPERB(NB)
      IF(NPER.GT.NPERMX) THEN
      WRITE(NFOUT,150) NB,NPER
 150  FORMAT(2X,'Block too big in SOLVER, NB and NPER : ',2I5)
      STOP
      ENDIF
C-----------------------------------------------------------------
C     Set counters and indices
C-----------------------------------------------------------------
      NPER=NPERB(NB)
      NPERSQ=NPER*NPER
      IF(NB.GT.1) IDSTRT=IDSTRT+NPERB(NB-1)
      IDEND=IDSTRT+NPER-1
      ILSTRT=IDEND+1
      NSL=NORD-IDEND
      NSU=IDSTRT-1
C-----------------------------------------------------------------
C     Loop over columns in block
C-----------------------------------------------------------------
      DO 300 JCOL=1,NPER
      NCOUNT=NCOUNT+1
      DO 175 I=1,NORD
 175  ATEM(I)=A((NCOUNT-1)*NORD+I)
C-----------------------------------------------------------------
C     Pick out diagonal block elements
C-----------------------------------------------------------------
      KST=(JCOL-1)*NPER
      DO 200 I=1,NPER
      KD=KST+I
      ID=I+IDSTRT-1
 200  AD(KD)=ATEM(ID)
C-----------------------------------------------------------------
C     Write column to lower triangular file
C-----------------------------------------------------------------
      IF(NB.EQ.NBLOCK) GO TO 220
      DO 2 J=1,NSL
      UABL(I1+J)=ATEM(ILSTRT+J-1)
 2    CONTINUE
      I1=I1+NSL
 220  CONTINUE
C-----------------------------------------------------------------
C     Write column to upper triangular file
C-----------------------------------------------------------------
      IF(NB.EQ.1) GO TO 240
      DO 3 J=1,NSU
 3    UABU(I2+J)=ATEM(J)
```

```
      I2=I2+NSU
 240  CONTINUE
 300  CONTINUE
C-----------------------------------------------------------------
C     Now that all columns of current block have been read,
C     factor diagonal block and write to diagonal file
C-----------------------------------------------------------------
      CALL DECOMP(NPER,NPER,AD,COND,IPVT,WORK)
      DO 4 J=1,NPERSQ
 4     UABD(I3+J)=AD(J)
      I3=I3+NPERSQ
      DO 5 J=1,NPER
 5     UABD(I3+J)=IPVT(J)
      I3=I3+NPER
 490  CONTINUE
 500  CONTINUE
C-----------------------------------------------------------------
C     Initialize variables to start iteration
C-----------------------------------------------------------------
      IT=0
      SUM=0.0
      DO 530 N=1,NORD
      DSIG(N)=0.0
      SIG(N)=0.0
      DUM=RHSM(N)
      RES(N)=DUM
 530  SUM=SUM+DUM*DUM
      P(1,1)=SUM
C-----------------------------------------------------------------
C     Write first solution and residual to file
C-----------------------------------------------------------------
      URS1(1)=IT
      DO 6 J=1,NORD
 6     URS1(1+J)=RES(J)
      DO 7 J=1,NORD
 7     URS1(NORD+1+J)=DSIG(J)
 1000 CONTINUE
C-----------------------------------------------------------------
C     Start blocked GAUSS-SIEDEL pass
C-----------------------------------------------------------------
      IDSTRT=1
      I1=0
      I2=0
      DO 1600 NB=1,NBLOCK
      NPER=NPERB(NB)
      NPERSQ=NPER*NPER
      IF(NB.GT.1) IDSTRT=IDSTRT+NPERB(NB-1)
      IDEND=IDSTRT+NPER-1
      ILSTRT=IDEND+1
      NSL=NORD-IDEND
      NSU=IDSTRT-1
C-----------------------------------------------------------------------
C     Read in diagonal block (already factored)
C-----------------------------------------------------------------------
      DO 8 J=1,NPERSQ
 8     AD(J)=UABD(I1+J)
      I1=I1+NPERSQ
      DO 9 J=1,NPER
      IPVT(J)=UABD(I1+J)
 9     CONTINUE
      I1=I1+NPER
C-----------------------------------------------------------------------
C     Set up diagonal block equations
C-----------------------------------------------------------------------
      DO 1050 IROW=1,NPER
      IROWM=IROW+IDSTRT-1
 1050 RHSD(IROW)=RES(IROWM)
C-----------------------------------------------------------------------
C     Solve diagonal block equations
C-----------------------------------------------------------------------
      CALL DSOLVE(NPER,NPER,AD,RHSD,IPVT)
      DO 1100 IROW=1,NPER
      IROWM=IROW+IDSTRT-1
 1100 DSIG(IROWM)=RHSD(IROW)
C-----------------------------------------------------------------------
C     Subtract lower triangular matrices from RES (residual)
C-----------------------------------------------------------------------
      IF (NB.EQ.NBLOCK) GO TO 1500
      DO 1400 JCOL=1,NPER
      JM=JCOL+IDSTRT-1
      DO 10 J=1,NSL
 10    ATEM(J)=UABL(I2+J)
      I2=I2+NSL
      DSIGM=DSIG(JM)
      DO 1200 IROW=1,NSL
      IM=IROW+ILSTRT-1
 1200 RES(IM)=RES(IM)-ATEM(IROW)*DSIGM
 1400 CONTINUE
 1500 CONTINUE
 1600 CONTINUE
```

```
C------------------------------------------------------------------
C    Update residual vector using upper triangular matrix blocks
C    and solution perturbation just obtained
C------------------------------------------------------------------
     IDSTRT=1
     DO 1620 N=1,NORD
 1620 RES(N)=0.0
     I1=0
     DO 1800 NB=1,NBLOCK
     NPER=NPERB(NB)
     IF(NB.GT.1) IDSTRT=IDSTRT+NPERB(NB-1)
     IDEND=IDSTRT+NPER-1
     ILSTRT=IDEND+1
     NSU=IDSTRT-1
     NSL=NORD-IDEND
     IF(NB.EQ.1) GO TO 1750
     DO 1700 JCOL=1,NPER
     DO 11 J=1,NSU
 11   ATEM(J)=UABU(I1+J)
     I1=I1+NSU
     JM=JCOL+IDSTRT-1
     DSIGM=DSIG(JM)
     DO 1650 IROW=1,NSU
     IM=IROW
 1650 RES(IM)=RES(IM)-ATEM(IM)*DSIGM
 1700 CONTINUE
 1750 IF(NB.EQ.NBLOCK) THEN
     DO 1770 IROW=1,NPER
     IROWM=IROW+IDSTRT-1
 1770 RES(IROWM)=0.0
     ENDIF
 1800 CONTINUE
C------------------------------------------------------------------
C    Increment iteration counter
C------------------------------------------------------------------
     IT=IT+1
C------------------------------------------------------------------
C    This finishes pass thru GAUSS-SIEDEL iteration
C    DSIG contains solution perturbation vector, and RES now contains
C    new residual vector
C
C    Now compute acceleration coefficients F
C
C    Find new diagonal element of P matrix (before acceleration) and
C    new solution (before acceleration)
```

```
C-----------------------------------------------------------------
     SUM=0.0
     DO 1900 N=1,NORD
     SIGNEW(N)=SIG(N)+DSIG(N)
     DUM=RES(N)
     RESNEW(N)=DUM
 1900 SUM=SUM+DUM**2
     P(IT+1,IT+1)=SUM
C-----------------------------------------------------------------
C   Now read through previous residual vectors
C   and complete P matrix (before acceleration)
C-----------------------------------------------------------------
     ITPREV=URS1(1)
     I1=0
     DO 2100 IRES=1,ITPREV+1
     DO 12 J=1,NORD
 12  ATEM(J)=URS1(I1+1+J)
     I1=I1+NORD
     SUM=0.0
     DO 2000 N=1,NORD
     SUM=SUM+ATEM(N)*RES(N)
 2000 CONTINUE
     P(IT+1,IRES)=SUM
     P(IRES,IT+1)=SUM
 2100 CONTINUE
C-----------------------------------------------------------------
C   Now compute Q-matrix elements from P-matrix elements
C-----------------------------------------------------------------
     DO 2200 I=1,IT
     DO 2150 J=1,IT
 2150 Q11(I,J)=P(I,J)-P(IT+1,I)-P(IT+1,J)+P(IT+1,IT+1)
     Q12(I)=P(IT+1,I)-P(IT+1,IT+1)
 2200 Q12(I)=-Q12(I)
C-----------------------------------------------------------------
C   Solve for acceleration coefficients
C-----------------------------------------------------------------
     CALL DECOMP(MAXDM,IT,Q11,COND,IPVT,WORK)
     CALL DSOLVE(MAXDM,IT,Q11,Q12,IPVT)
     SUM=0.0
     DO 2350 I=1,IT
     F(I)=Q12(I)
 2350 SUM=SUM+F(I)
     F(IT+1)=1.0-SUM
C-----------------------------------------------------------------
C   Now recompute latest solution and residual vectors
```

```
C-----------------------------------------------------------------------
      DO 2400 N=1,NORD
      SIG(N)=0.0
 2400 RES(N)=0.0
C-----------------------------------------------------------------------
C     Put together new residual vector and write residuals to URS2
C-----------------------------------------------------------------------
      I1=0
      I2=0
      ITPREV=URS1(1)
      URS2(1)=IT
      I1=I1+1
      I2=I2+1
      DO 2500 ITC=1,ITPREV+1
      FMULT=F(ITC)
      DO 13 J=1,NORD
 13   ATEM(J)=URS1(I1+J)
      I1=I1+NORD
      DO 14 J=1,NORD
 14   URS2(I2+J)=ATEM(J)
      I2=I2+NORD
      DO 2450 I=1,NORD
 2450 RES(I)=RES(I)+FMULT*ATEM(I)
 2500 CONTINUE
      RESERR=0.0
      FMULT=F(IT+1)
      DO 2550 I=1,NORD
      RES(I)=RES(I)+FMULT*RESNEW(I)
      RESERR=AMAX1(RESERR,ABS(RES(I)))
 2550 CONTINUE
      DO 15 J=1,NORD
 15   URS2(I2+J)=RES(J)
      I2=I2+NORD
C-----------------------------------------------------------------------
C     Put together new solution vector and write solutions to URS2
C-----------------------------------------------------------------------
      DO 2700 ITC=1,ITPREV+1
      FMULT=F(ITC)
      DO 16 J=1,NORD
 16   ATEM(J)=URS1(I1+J)
      I1=I1+NORD
      DO 17 J=1,NORD
 17   URS2(I2+J)=ATEM(J)
      I2=I2+NORD
      DO 2650 I=1,NORD
 2650 SIG(I)=SIG(I)+FMULT*ATEM(I)
 2700 CONTINUE
      FMULT=F(IT+1)
      DO 2750 I=1,NORD
      SIG(I)=SIG(I)+FMULT*SIGNEW(I)
 2750 CONTINUE
      DO 18 J=1,NORD
      URS2(I2+J)=SIG(J)
  18  CONTINUE
      I2=I2+NORD
C-----------------------------------------------------------------
C    Check for convergence and exit if satisfied
C-----------------------------------------------------------------
      IF(RESERR.LE.TOL) THEN
      WRITE(NFOUT,2780) IT
 2780 FORMAT(2X,'CONVERGED AT ITERATION # : ',I2)
      RETURN
      ENDIF
      IF(IT.GE.ITMAX) THEN
      WRITE(NFOUT,2800) ITMAX,RESERR
 2800 FORMAT(/2X,'NO CONVERGENCE IN SOLVER AFTER',I4,
     &       ' ITERATIONS',/2X,'MAX RESIDUAL=',1PE10.4)
      RETURN
      ENDIF
C-----------------------------------------------------------------
C    Recompute final entries in matrix P
C    All elements in final row are equal
C-----------------------------------------------------------------
      SUM=0.0
      DO 2900 ITC=1,IT+1
 2900 SUM=SUM+F(ITC)*P(ITC.IT+1)
      DO 2930 ITC=1,IT+1
      P(IT+1,ITC)=SUM
      P(ITC,IT+1)=SUM
 2930 CONTINUE
C-----------------------------------------------------------------
C    Swap acceleration arrays
C-----------------------------------------------------------------
      IMAX=MAX0(I1,I2)
      DO 19 J=1,IMAX
      TEMP=URS1(J)
      URS1(J)=URS2(J)
      URS2(J)=TEMP
  19  CONTINUE
C-----------------------------------------------------------------
```

```
C    Zero solution perturbation vector
C------------------------------------------------------------------
      DO 3000 N=1,NORD
 3000 DSIG(N)=0.0
C------------------------------------------------------------------
C    Go back for another block iteration pass
C------------------------------------------------------------------
      GO TO 1000
      END


      SUBROUTINE DSOLVE(NDIM,N,A,B,IPVT)
C------------------------------------------------------------------
C    This subroutine solves the diagonal block equations
C------------------------------------------------------------------
      INTEGER IPVT(N)
      REAL A(NDIM.N),B(N)
      IF(N.EQ.1) GO TO 50
      NM1=N-1
      DO 20 K=1,NM1
      KP1=K+1
      M=IPVT(K)
      T=B(M)
      B(M)=B(K)
      B(K)=T
      DO 10 I=KP1,N
      B(I)=B(I)+A(I,K)*T
   10 CONTINUE
   20 CONTINUE
      DO 40 KB=1,NM1
      KM1=N-KB
      K=KM1+1
      B(K)=B(K)/A(K,K)
      T=-B(K)
      DO 30 I=1,KM1
      B(I)=B(I)+A(I,K)*T
   30 CONTINUE
   40 CONTINUE
   50 B(1)=B(1)/A(1,1)
C------------------------------------------------------------------
C    Exit
C------------------------------------------------------------------
      RETURN
      END
```

Appendix H

```
      SUBROUTINE UGLYDK(NIN,NCL,NCR,XIN,YIN,ESL,ESR,AE,ITS)
C-----1975 DUCK SERIES J.E.KERWIN  MODIFIED 6/21/82---------------------
C-----TRI-DIAGONAL MATRIX SOULTION BUILT IN----------------------------
C     MAY 14, 1983 UPDATE J. H. MILGRAM
      DIMENSION XIN(1),YIN(1),AE(1),H(50),D(50),AU(50),AM(50),S(50),
     *          AL(50),X(50)
      DATA HALF/0.5E00/.TWO/2.0E00/,SIX/6.0E00/,RAD/1.745329E-02/
      ITS=0
      IF (NCR.EQ.4) ITS=2
      IF (NCL.EQ.4) ITS=ITS+1
      NM1=NIN-1
      NM2=NM1-1
      NM3=NM2-1
      NEQ=NM2
      DO 1 N=1,NM1
      H(N)=XIN(N+1)-XIN(N)
 1    D(N)=(YIN(N+1)-YIN(N))/H(N)
      IF(NCL.EQ.2) NEQ=NEQ+1
      IF(NCR.EQ.2) NEQ=NEQ+1
      NSQ=NEQ**2
      J=1
      IF(NCL.NE.2) GO TO 6
      AM(1)=TWO*H(1)
      AU(1)=H(1)
      SLP=ESL*RAD
      S(1)=(D(1)-TAN(SLP))*SIX
      J=J+1
      AL(2)=H(1)
6     N=1
```

```
C     IF (NM2.EQ.1) GO TO 70
      DO 5 N=1,NM2
70    IF(N.GT.1) AU(J-1)=H(N)
      AM(J)=TWO*(H(N)+H(N+1))
      IF(N.LT.NM2) AL(J+1)=H(N+1)
      IF(N.EQ.1.AND.NCL.EQ.3)AM(J)=AM(J)+H(N)
      IF(N.EQ.1.AND.NCL.EQ.4)AM(J)=AM(J)+10.*H(N)
      IF(N.EQ.NM2.AND.NCR.EQ.3)AM(J)=AM(J)+H(N)
      IF(N.EQ.NM2.AND.NCR.EQ.4)AM(J)=AM(J)+10.*H(N)
      IF(N.EQ.2.AND.NCL.EQ.1) AU(J-1)=AU(J-1)-H(N-1)**2/H(N)
      IF(N.EQ.1.AND.NCL.EQ.1) AM(J)=AM(J)+(1.0+H(N)/H(N+1))*H(N)
      IF(N.EQ.NM2.AND.NCR.EQ.1) AM(J)=AM(J)+(1.0+H(N+1)/H(N))*H(N+1)
      IF(N.EQ.NM3.AND.NCR.EQ.1) AL(J+1)=AL(J+1)-H(N+2)**2/H(N+1)
      S(J)=(D(N+1)-D(N))*SIX
      J=J+1
 5    CONTINUE
      IF(NCR.NE.2) GO TO 7
      AL(NEQ)=-H(NM1)
      AM(NEQ)=-TWO*H(NM1)
      AU(NEQ-1)=H(NM1)
      SLP=ESR*RAD
      S(J)=(D(NM1)+TAN(SLP))*SIX
 7    CONTINUE
      K=2
C     IF (NEQ.EQ.2) GO TO 50
      DO 4 K=2,NEQ
50    AL(K)=AL(K)/AM(K-1)
      AM(K)=AM(K)-AL(K)*AU(K-1)
      S(K)=S(K)-AL(K)*S(K-1)
 4    CONTINUE
```

```
      X(NEQ)=S(NEQ)/AM(NEQ)
      L=2
C      IF (NEQ.EQ.2) GO TO 60
      DO 2 L=2,NEQ
60     K=NEQ-L+1
      IF (K.LT.1) K=1
      X(K)=(S(K)-AU(K)*X(K+1))/AM(K)
 2    CONTINUE
      N=1
C      IF (NEQ.EQ.1) GO TO 22
      DO 22 N=1,NEQ
 22    S(N)=X(N)
      HOLD=S(NEQ)
      IF(NCL.EQ.2) GO TO 8
      N=1
      IF (NM2.EQ.1) GO TO 23
      DO 9 N=1,NM2
23     M=NM2-N+2
 9     S(M)=S(M-1)
      IF(NCL.EQ.0) S(1)=0.0
      IF(NCL.EQ.3)S(1)=S(2)
      IF (NCL.EQ.-1) S(1)=ESL*S(2)
      BUG=H(1)/H(2)
      IF(NCL.EQ.1) S(1)=(1.0+BUG)*S(2)-BUG*S(3)
 8     IF(NCR.EQ.0) S(NIN)=0.0
      IF(NCR.EQ.3)S(NIN)=S(NM1)
      IF (NCR.EQ.-1) S(NIN)=ESR*S(NM1)
      BUG=H(NM1)/H(NM2)
      IF(NCR.EQ.1) S(NIN)=(1.0+BUG)*S(NM1)-BUG*S(NM2)
      IF(NCR.EQ.2) S(NIN)=HOLD
      DO 10 N=1,NM1
      L=4*(N-1)+1
      IF(N.NE.1.OR.NCL.NE.4)GO TO 30
      AE(2)=-4.*S(2)*H(1)**1.5
      AE(3)=D(1)+4.*S(2)*H(1)
      AE(4)=YIN(1)
      AE(1)=0.0
      GO TO 14
30    CONTINUE
      IF(N.NE.NM1.OR.NCR.NE.4)GO TO 40
      AE(L+1)=-4.*S(NM1)*H(NM1)**1.5
      AE(L+2)=D(NM1)-4.*S(NM1)*H(NM1)
      AE(L+3)=YIN(NM1)+4.*S(NM1)*H(NM1)**2
      AE(L)=0.0
      GO TO 14
40    AE(L)=(S(N+1)-S(N))/(SIX*H(N))
      M=L+1
      AE(M)=HALF*S(N)
      M=M+1
      AE(M)=D(N)-H(N)*(TWO*S(N)+S(N+1))/SIX
      M=M+1
      AE(M)=YIN(N)
14    CONTINUE
10    CONTINUE
      RETURN
      END
      SUBROUTINE EVALDK(NIN,NOUT,XIN,XOUT,YOUT,A,ITS)
C     APRIL 1975 SPLINE PROGRAM SERIES   J.E.KERWIN
C     MAY 14, 1983 UPDATE J. H. MILGRAM
      DIMENSION XIN(1),XOUT(1),YOUT(1),A(1)
      NM1=NIN-1
      MOUT=IABS(NOUT)
      IF(NOUT.GT.0) GO TO 1
      DEL=(XIN(NIN)-XIN(1))/(MOUT-1)
      N=1
C      IF (MOUT.EQ.1) GO TO 2
      DO 2 N=1,MOUT
 2    XOUT(N)=XIN(1)+(N-1)*DEL
 1    J=1
      N=1
C     IF (MOUT.EQ.1) GO TO 20
      DO 3 N=1,MOUT
20     IF(XOUT(N).GE.XIN(2)) GO TO 4
      J=1
      GO TO 5
 4    IF(XOUT(N).LT.XIN(NM1)) GO TO 6
      J=NM1
      GO TO 5
 6    IF(XOUT(N).GE.XIN(J+1)) GO TO 7
    9 IF (XOUT(N).LT.XIN(J)) GO TO 8
 5    H1=XOUT(N)-XIN(J)
      H2=H1**2
      H3=H1*H2
      J1=4*(J-1)+1
      J2=J1+1
      J3=J2+1
      J4=J3+1
      IF((ITS.EQ.0).OR.(J.NE.1.AND.J.NE.NM1))GO TO 10
      IF(ITS.EQ.1.AND.J.NE.1)GO TO 10
      IF(ITS.EQ.2.AND.J.NE.NM1) GO TO 10
      IF(J.EQ.1)YOUT(N)=A(J2)*SQRT(ABS(H1))+A(J3)*H1+A(J4)
      htest = xin(nin) -xout(n)
      IF(J.EQ.NM1)YOUT(N)=A(J2)*SQRT(ABS(htest)   )+A(J3)*H1+A(J4)
      GO TO 3
10     YOUT(N)=A(J1)*H3+A(J2)*H2+A(J3)*H1+A(J4)
      GO TO 3
7     J=J+1
      GO TO 6
    8 J=J-1
      GO TO 9
3     CONTINUE
      RETURN
      END
      SUBROUTINE INTEDK(NIN,XIN,XL,XU,YDX,XYDX,XXYDX,A,ITS)
C     AUGUST 1975 SPLINE PROGRAM SERIES  S.-K.TSAO
C     MAY 14, 1983 UPDATE J. H. MILGRAM
      DIMENSION XIN(1),A(1)
      BA(X1,X2)=AA*A(2)*((X2-XS)**1.5-(X1-XS)**1.5)+0.5*A(3)*
     1((X2-XS)**2-(X1-XS)**2)+A(4)*(X2-X1)
      BM(X1,X2)=0.4*A(2)*((X2-XS)**2.5-(X1-XS)**2.5)+A(3)*((X2-XS)
     1**3-(X1-XS)**3)/3.+0.5*A(4)*((X2-XS)**2-(X1-XS)**2)+XS*BA(X1,X2)
      BI(X1,X2)=AB*A(2)*((X2-XS)**3.5-(X1-XS)**3.5)+0.25*A(3)*((X2-XS)
     1**4-(X1-XS)**4)+A(4)*((X2-XS)**3-(X1-XS)**3)/3.+
     12*XS*BM(X1,X2)-XS*XS*BA(X1,X2)
      EA1(X1,X2)=AA*A(I2)*((XR-X1)**1.5-(XR-X2)**1.5)
      EA2(X1,X2)=0.5*A(I3)*((X2-X9)**2-(X1-X9)**2)+A(I4)*(X2-X1)
      EA(X1,X2)=EA1(X1,X2)+EA2(X1,X2)
      EB1(X1,X2)=0.5*A(I2)*((XR-X2)**2.5-(XR-X1)**2.5)
      EB2(X1,X2)=A(I3)*((X2-X9)**3-(X1-X9)**3)/3.+0.5*A(I4)*
     1((X2-X9)**2-(X1-X9)**2)
```

```
      EM1(X1,X2)=XR*EA1(X1.X2)+EB1(X1,X2)
      EM2(X1,X2)=X9*EA2(X1.X2)+EB2(X1,X2)
      EM(X1,X2)=EM1(X1,X2)+EM2(X1,X2)
      EI1(X1,X2)=AB*A(I2)*((XR-X1)**3.5-(XR-X2)**3.5)
     1+2.*XR*EM1(X1,X2)-XR*XR*EA1(X1,X2)
      EI2(X1,X2)=0.25*A(I3)*((X2-X9)**4-(X1-X9)**4)+A(I4)*
     1((X2-X9)**3-(X1-X9)**3)/3.+2.*X9*EM2(X1,X2)-X9*X9*EA2(X1,X2)
      EI(X1,X2)=EI1(X1,X2)+EI2(X1,X2)
      AA=2./3.
      AB=2./7.
      NM1=NIN-1
      I1=4*(NM1-1)+1
      I2=I1+1
      I3=I2+1
      I4=I3+1
      X9=XIN(NM1)
      XS=XIN(1)
      XR=XIN(NIN)
      IF(XL.LT.XIN(1)) GO TO 2
      DO 1 N=1,NIN
      IF(XL.GE.XIN(N)) GO TO 1
      JL=N-1
      GO TO 3
 1    CONTINUE
      GO TO 3
 2    JL=1
 3    IF(XU.GE.XIN(NIN)) GO TO 4
      JU=1
      IF(XU.LE.XIN(1)) GO TO 6
      DO 5 N=JL,NIN
```

```
  IF(XU.GE.XIN(N)) GO TO 5
  JU=N-1
  GO TO 6
5 CONTINUE
  GO TO 6
4 JU=NM1
6 H1=XL-XIN(JL)
  IF((JL.EQ.1).AND.(ITS.EQ.1.OR.ITS.EQ.3))GO TO 10
  IF((JL.EQ.NM1).AND.(ITS.EQ.2.OR.ITS.EQ.3))GO TO 15
  H2=H1**2
  H3=H1*H2
  H4=H2**2
  H5=H2*H3
  H6=H3**2
  J1=4*(JL-1)+1
  J2=J1+1
  J3=J2+1
  J4=J3+1
  YDX=-A(J1)/4.0*H4-A(J2)/3.0*H3-A(J3)/2.0*H2-A(J4)*H1
  BUG=-A(J1)/5.0*H5-A(J2)/4.0*H4-A(J3)/3.0*H3-A(J4)/2.0*H2
  XYDX=BUG+XIN(JL)*YDX
  BUG=-A(J1)/6.0*H6-A(J2)/5.0*H5-A(J3)/4.0*H4-A(J4)/3.0*H3
  XXYDX=BUG+2.0*XIN(JL)*XYDX-XIN(JL)**2*YDX
  GO TO 20
10  XL1=XL
  IF((XL1-XS).LT.0.0) XL1=XS+1.E-6
  YDX=-BA(XS,XL1)
  XYDX=-BM(XS,XL1)
  XXYDX=-BI(XS,XL1)
  GO TO 20
```

```
15   XL1=XL
     IF((XR-XL1).LT.0.0) XL1=XR-1.E-6
     YDX=EA(X9.XL1)
     XYDX=EM(X9,XL1)
     XXYDX=EI(X9,XL1)
20   CONTINUE
     N=JL
C    IF (JL.EQ.JU) GO TO 22
     DO 7 N=JL,JU
22   X1=XIN(N)
     X2=XIN(N+1)
     IF(N.EQ.JU) X2=XU
     IF(N.EQ.1.AND.(ITS.EQ.1.OR.ITS.EQ.3))GO TO 25
     IF(N.EQ.NM1.AND.(ITS.EQ.2 .OR.ITS.EQ.3)) GO TO 30
     H1=X2-X1
     H2=H1**2
     H3=H1*H2
     H4=H2**2
     H5=H2*H3
     H6=H3**2
     J1=4*(N-1)+1
     J2=J1+1
     J3=J2+1
     J4=J3+1
     IF(N.EQ.NM1.AND.(ITS.EQ.2.OR.ITS.EQ.3))GO TO 30
     BUG=A(J1)/4.0*H4+A(J2)/3.0*H3+A(J3)/2.0*H2+A(J4)*H1
     YDX=YDX+BUG
     CAT=A(J1)/5.0*H5+A(J2)/4.0*H4+A(J3)/3.0*H3+A(J4)/2.0*H2
     PIG=CAT+XIN(N)*BUG
     XYDX=XYDX+PIG
```

```
      DOG=A(J1)/6.0*H6+A(J2)/5.0*H5+A(J3)/4.0*H4+A(J4)/3.0*H3
      XXYDX=XXYDX+DOG+2.0*XIN(N)*PIG-XIN(N)**2*BUG
      GO TO 7
25    CONTINUE
      IF(X2.GT.XR)X2=XR
      IF(X2.LT.XS) X2=XS
      YDX=YDX+BA(X1,X2)
      XYDX=XYDX+BM(X1,X2)
      XXYDX=XXYDX+BI(X1,X2)
      GO TO 7
30    CONTINUE
      IF(X2.GT.XR) X2=XR
      IF(X2.LT.XS) X2=XS
      YDX=YDX+EA(X1,X2)
      XYDX=XYDX+EM(X1,X2)
      XXYDX=XXYDX+EI(X1,X2)
 7    CONTINUE
      RETURN
      END
      FUNCTION FILLIN(X,AB,OR,NO)
C     *** FILLIN *** PARABOLIC INTERPOLATION
C     FIND Y(X) FROM TABLE OF
C     AB(N) AND OR(N) CONTAINING NO POINTS.
      DIMENSION AB(2),OR(2)
      ANTRA(X1,X2,X3,X,Y1,Y2,Y3)=Y1*(X-X2)*(X-X3)/((X1-X2)*(X1-X3))+
     1 Y2*(X-X1)*(X-X3)/((X2-X1)*(X2-X3))+Y3*(X-X1)*(X-X2)/((X3-X1)*
     2 (X3-X2))
      IF(X-AB(1))  1,3,2
 3    Y=OR(1)
      GO TO 99
```

```
1   Y=ANTRA(AB(1),AB(2),AB(3).X,OR(1),OR(2),OR(3))
    GO TO 99
2   IF(X-AB(2))1,6,5
6   Y=OR(2)
    GO TO 99
5   DO 7 I=3,NO
    M=I
    IF(X-AB(I))8,9,7
9   Y=OR(I)
    GO TO 99
7   CONTINUE
8   Y=ANTRA(AB(M-2),AB(M-1).AB(M),X,OR(M-2),OR(M-1),OR(M))
99  FILLIN=Y
    RETURN
    END
    SUBROUTINE DRIVDK(NIN,NOUT,XIN,XOUT,DYDX,D2YDX,A,ITS)
C    APRIL 1975 SPLINE PROGRAM SERIES   J.E.KERWIN
C    MAY 14, 1983 UPDATE J. H. MILGRAM
    DIMENSION XIN(1),XOUT(1),DYDX(1),D2YDX(1),A(1)
    NM1=NIN-1
    J=1
    DO 3 N=1,NOUT
    IF(XOUT(N).GE.XIN(2)) GO TO 4
    J=1
    GO TO 5
4   IF(XOUT(N).LT.XIN(NM1)) GO TO 6
    J=NM1
    GO TO 5
6   IF(XOUT(N).GE.XIN(J+1)) GO TO 7
5   H1=XOUT(N)-XIN(J)
```

```
      H2=H1**2
      J1=4*(J-1)+1
      J2=J1+1
      J3=J2+1
      IF(ITS.EQ.0) GO TO 10
      IF(J.NE.1.AND.J.NE.NM1)GO TO 10
      IF((J.EQ.1).AND.(ITS.EQ.2)) GO TO 10
      IF((J.EQ.NM1).AND.(ITS.EQ.1)) GO TO 10
      IF(J-1)20,20,30
20     F=SQRT(ABS(H1))
      IF (ABS(F).LT.1.E-10) F=1.E-10
      DYDX(N)=0.5*A(J2)/F+A(J3)
      D2YDX(N)=-0.25*A(J2)/(F*H1)
      GO TO 3
30     F=SQRT(XIN(NIN)-XOUT(N))
      IF(F.LT.1.E-10)F=1.E-10
      DYDX(N)=-0.5*A(J2)/F+A(J3)
      D2YDX(N)=-0.25*A(J2)/F**3
      GO TO 3
10     DYDX(N)=3.0*A(J1)*H2+2.0*A(J2)*H1+A(J3)
      D2YDX(N)=6.0*A(J1)*H1+2.0*A(J2)
      GO TO 3
7     J=J+1
      GO TO 6
3     CONTINUE
      RETURN
      END
C
C
C
```

```
      SUBROUTINE LINDK(NIN,XIN,YIN,A)
C-----GENERATE CUBIC COEFFICIENTS FOR PIECEWISE LINEAR FIT-------------
-
C     MAY 14, 1983 UPDATE CORRECTING COEFFICIENT SEQUENCE
C //
      DIMENSION XIN(1),YIN(1),A(1)
      NM1=NIN-1
      DO 1 N=1,NM1
      L=4*(N-1)+1
      A(L)=0.0
      M=L+1
      A(M)=0.0
      M=M+1
      A(M)=(YIN(N+1)-YIN(N))/(XIN(N+1)-XIN(N))
      M=M+1
 1    A(M)=YIN(N)
      RETURN
      END
      SUBROUTINE SORT(NX,X,Y)
      DIMENSION X(NX),Y(NX)
      DO 20 NEXT=1,NX
      XLOW=X(NEXT)
      NLOW=NEXT
      DO 10 N=NEXT,NX
      IF(X(N).GE.XLOW)GO TO 10
      NLOW=N
      XLOW=X(N)
  10  CONTINUE
      IF(NEXT.EQ.NLOW)GO TO 20
      TEMP=X(NEXT)
      X(NEXT)=X(NLOW)
      X(NLOW)=TEMP
      TEMP=Y(NEXT)
      Y(NEXT)=Y(NLOW)
      Y(NLOW)=TEMP
   20 CONTINUE
      RETURN
      END
      SUBROUTINE LLSQ(A,B,M,N,L,X,IPIV,EPS,IER,AUX)
C     THE ABOVE CARD SHOULD BE PLACED IN PROPER SEQUENCE
C        BEFORE COMPILING THIS UNDER  IBM FORTRAN G.
C
C     ..................................................................
C
C        SUBROUTINE LLSQ
C
C        PURPOSE
C           TO SOLVE LINEAR LEAST SQUARES PROBLEMS, I.E. TO MINIMIZE
C           THE EUCLIDEAN NORM OF B-A*X, WHERE A IS A M BY N MATRIX
C           WITH M NOT LESS THAN N. IN THE SPECIAL CASE M=N SYSTEMS OF
C           LINEAR EQUATIONS MAY BE SOLVED.
C
C        USAGE
C           CALL LLSQ (A,B,M,N,L,X,IPIV,EPS,IER,AUX)
C
C        DESCRIPTION OF PARAMETERS
C           A      - M BY N COEFFICIENT MATRIX (DESTROYED).
C           B      - M BY L RIGHT HAND SIDE MATRIX (DESTROYED).
C           M      - ROW NUMBER OF MATRICES A AND B.
```

```
C        N     - COLUMN NUMBER OF MATRIX A, ROW NUMBER OF MATRIX
X.
C        L     - COLUMN NUMBER OF MATRICES B AND X.
C        X     - N BY L SOLUTION MATRIX.
C        IPIV  - INTEGER OUTPUT VECTOR OF DIMENSION N WHICH
C                CONTAINS INFORMATIONS ON COLUMN INTERCHANGES
C                IN MATRIX A. (SEE REMARK NO.3).
C        EPS   - INPUT PARAMETER WHICH SPECIFIES A RELATIVE
C                TOLERANCE FOR DETERMINATION OF RANK OF MATRIX A.
C        IER   - A RESULTING ERROR PARAMETER.
C        AUX   - AUXILIARY STORAGE ARRAY OF DIMENSION MAX(2*N,L).
C                ON RETURN FIRST L LOCATIONS OF AUX CONTAIN THE
C                RESULTING LEAST SQUARES.
C
C     REMARKS
C        (1) NO ACTION BESIDES ERROR MESSAGE IER=-2 IN CASE
C            M LESS THAN N.
C        (2) NO ACTION BESIDES ERROR MESSAGE IER=-1 IN CASE
C            OF A ZERO-MATRIX A.
C        (3) IF RANK K OF MATRIX A IS FOUND TO BE LESS THAN N BUT
C            GREATER THAN 0, THE PROCEDURE RETURNS WITH ERROR CODE
C            IER=K INTO CALLING PROGRAM. THE LAST N-K ELEMENTS OF
C            VECTOR IPIV DENOTE THE USELESS COLUMNS IN MATRIX A.
C            THE REMAINING USEFUL COLUMNS FORM A BASE OF MATRIX A.
C        (4) IF THE PROCEDURE WAS SUCCESSFUL, ERROR PARAMETER IER
C            IS SET TO 0.
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C        NONE
C
```

```
C    METHOD
C      HOUSEHOLDER TRANSFORMATIONS ARE USED TO TRANSFORM
MATRIX A
C      TO UPPER TRIANGULAR FORM. AFTER HAVING APPLIED THE SAME
C      TRANSFORMATION TO THE RIGHT HAND SIDE MATRIX B, AN
C      APPROXIMATE SOLUTION OF THE PROBLEM IS COMPUTED BY
C      BACK SUBSTITUTION. FOR REFERENCE, SEE
C      G. GOLUB, NUMERICAL METHODS FOR SOLVING LINEAR LEAST
C      SQUARES PROBLEMS, NUMERISCHE MATHEMATIK, VOL.7,
C      ISS.3 (1965), PP.206-216.
C
C  ..............................................................
C
C
   DIMENSION A(1),B(1),X(1),IPIV(1),AUX(1)
C
C  ERROR TEST
   IF(M-N)30,1,1
C
C  GENERATION OF INITIAL VECTOR S(K) (K=1,2,...,N) IN STORAGE
C  LOCATIONS AUX(K) (K=1,2,...,N)
  1 PIV=0.
   IEND=0
   DO 4 K=1,N
   IPIV(K)=K
   H=0.
   IST=IEND+1
   IEND=IEND+M
   DO 2 I=IST,IEND
  2 H=H+A(I)*A(I)
```

```
   AUX(K)=H
   IF(H-PIV)4,4,3
 3 PIV=H
   KPIV=K
 4 CONTINUE
C
C   ERROR TEST
   IF(PIV)31,31,5
C
C   DEFINE TOLERANCE FOR CHECKING RANK OF A
 5 SIG=SQRT(PIV)
   TOL=SIG*ABS(EPS)
C
C
C   DECOMPOSITION LOOP
   LM=L*M
   IST=-M
   DO 21 K=1,N
   IST=IST+M+1
   IEND=IST+M-K
   I=KPIV-K
   IF(I)8,8,6
C
C   INTERCHANGE K-TH COLUMN OF A WITH KPIV-TH IN CASE KPIV.GT.K
 6 H=AUX(K)
   AUX(K)=AUX(KPIV)
   AUX(KPIV)=H
   ID=I*M
   DO 7 I=IST,IEND
   J=I+ID
```

```
      H=A(I)
      A(I)=A(J)
    7 A(J)=H
C
C     COMPUTATION OF PARAMETER SIG
    8 IF(K-1)11,11,9
    9 SIG=0.
      DO 10 I=IST,IEND
   10 SIG=SIG+A(I)*A(I)
      SIG=SQRT(SIG)
C
C     TEST ON SINGULARITY
      IF(SIG-TOL)32,32,11
C
C     GENERATE CORRECT SIGN OF PARAMETER SIG
   11 H=A(IST)
      IF(H)12,13,13
   12 SIG=-SIG
C
C     SAVE INTERCHANGE INFORMATION
   13 IPIV(KPIV)=IPIV(K)
      IPIV(K)=KPIV
C
C     GENERATION OF VECTOR UK IN K-TH COLUMN OF MATRIX A AND OF
C     PARAMETER BETA
      BETA=H+SIG
      A(IST)=BETA
      BETA=1./(SIG*BETA)
      J=N+K
      AUX(J)=-SIG
    IF(K-N)14,19,19
C
C   TRANSFORMATION OF MATRIX A
  14 PIV=0.
    ID=0
    JST=K+1
    KPIV=JST
    DO 18 J=JST,N
    ID=ID+M
    H=0.
    DO 15 I=IST,IEND
    II=I+ID
  15 H=H+A(I)*A(II)
    H=BETA*H
    DO 16 I=IST,IEND
    II=I+ID
  16 A(II)=A(II)-A(I)*H
C
C   UPDATING OF ELEMENT S(J) STORED IN LOCATION AUX(J)
    II=IST+ID
    H=AUX(J)-A(II)*A(II)
    AUX(J)=H
    IF(H-PIV)18,18,17
  17 PIV=H
    KPIV=J
  18 CONTINUE
C
C   TRANSFORMATION OF RIGHT HAND SIDE MATRIX B
  19 DO 21 J=K,LM,M
    H=0.
```

```
      IEND=J+M-K
      II=IST
      DO 20 I=J,IEND
      H=H+A(II)*B(I)
   20 II=II+1
      H=BETA*H
      II=IST
      DO 21 I=J,IEND
      B(I)=B(I)-A(II)*H
   21 II=II+1
C     END OF DECOMPOSITION LOOP
C
C
C     BACK SUBSTITUTION AND BACK INTERCHANGE
      IER=0
      I=N
      LN=L*N
      PIV=1./AUX(2*N)
      DO 22 K=N,LN,N
      X(K)=PIV*B(I)
   22 I=I+M
      IF(N-1)26,26,23
   23 JST=(N-1)*M+N
      DO 25 J=2,N
      JST=JST-M-1
      K=N+N+1-J
      PIV=1./AUX(K)
      KST=K-N
      ID=IPIV(KST)-KST
      IST=2-J
   DO 25 K=1,L
   H=B(KST)
   IST=IST+N
   IEND=IST+J-2
   II=JST
   DO 24 I=IST,IEND
   II=II+M
 24 H=H-A(II)*X(I)
   I=IST-1
   II=I+ID
   IF (II .GT. I) X(I) = X(II)
   X(II)=PIV*H
 25 KST=KST+M
C
C
C    COMPUTATION OF LEAST SQUARES
 26 IST=N+1
   IEND=0
   DO 29 J=1,L
   IEND=IEND+M
   H=0.
   IF(M-N)29,29,27
 27 DO 28 I=IST,IEND
 28 H=H+B(I)*B(I)
   IST=IST+M
 29 AUX(J)=H
   RETURN
C
C    ERROR RETURN IN CASE M LESS THAN N
 30 IER=-2
```

```
   RETURN
C
C    ERROR RETURN IN CASE OF ZERO-MATRIX A
  31 IER=-1
   RETURN
C
C    ERROR RETURN IN CASE OF RANK OF MATRIX A LESS THAN N
  32 IER=K-1
   RETURN
   END
   SUBROUTINE
SM3DK(NP,NFR,NCLL,NCRR,ND,XP,YP.ESL,ESR,XD,AF,ITS,IER,
   1M)
C    APRIL 1975 SPLINE PROGRAM SERIES   J.E.KERWIN
C     ADJUSTED TO HOLD VALUES AT ENDPOINTS
C    ALLOWS INSERTION OF 1,2 OR 3 DUCKS AT LOWER X END.
C    ND IS THE TOTAL NUMBER OF DUCKS THAT RESULT. FIRST EXTRA
DUCK
C    IS BETWEEN PTS. 1 AND 2, SECOND BETWEEN 2 AND 3. FOR 3 EXTRA
C    THERE ARE 2 BETWEEN 1 AND 2 AND ONE BETWEEN 2 AND 3
C    JUNE 5, 1983 UPDATE  J. H. MILGRAM
    DIMENSION XP(1),YP(1),AF(1),C(20),YQ(800),IPIV(40),AUX(40),YSP(100
   1 ),XD(1),YD(18),YY(100),AE(68,20),AG(68),XF(100),YF(100),AA(800)
    DIMENSION AS(100),SCP(100)
    M2=M/2
    M1=M-M2
    DO 100 I=1.100
    AS(I)=0.0
100  SCP(I)=0.0
    NP1=NP-1
```

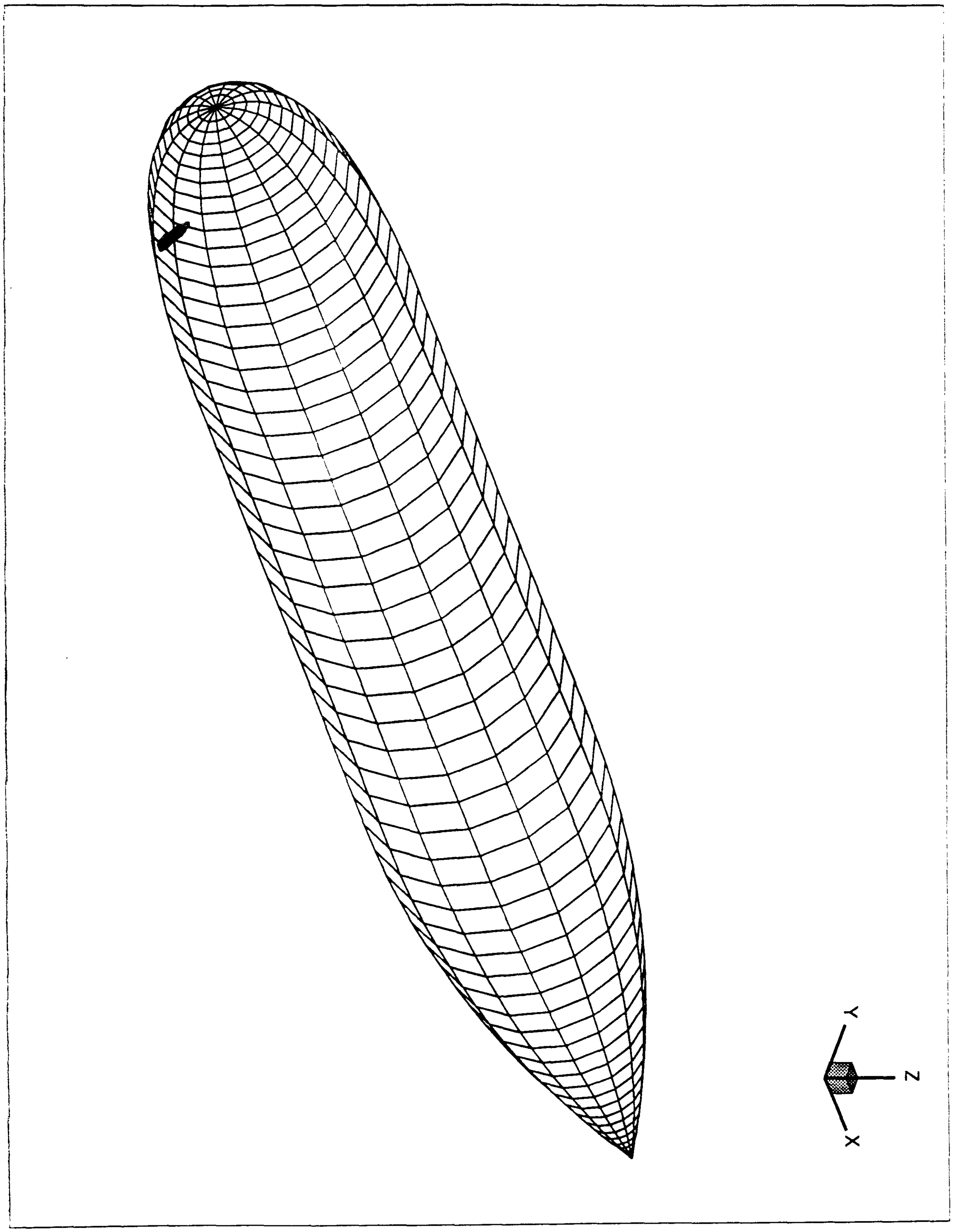
```
      SLP=(YP(NP)-YP(1))/(XP(NP)-XP(1))
      YBG=YP(1)
      XBG=XP(1)
      DO 20 I=1,NP
 20    YSP(I)=YP(I)-YBG-SLP*(XP(I)-XBG)
      ND2=ND-2
      NF=IABS(NFR)
      NMX=ND2*NF
      IF(NFR.LT.0) GO TO 10
      IF(NF.EQ.NP) GO TO 6
      DEL=(XP(NP)-XP(1))/(NF-1)
      DO 7 N=1,NF
  7    XF(N)=XP(1)+(N-1)*DEL
      GO TO 8
  6    DO 9 N=1,NP
      XF(N)=XP(N)
  9    YF(N)=YSP(N)
  8    DEL=(XP(NP)-XP(1))/(ND-M-1)
      IQ=0
      NDM=ND-M
      DO 1 N=1,NDM
      IQ=IQ+1
      XD(IQ)=(N-1)*DEL+XP(1)
      IF (IQ.EQ.1)IQ=IQ+M1
  1    IF (IQ.EQ.(2+M1)) IQ=IQ+M2
      XD(ND)=XP(NP)
      IF (M1.NE.1) GO TO 70
      XD(2)=XD(1)+DEL/2.
 70    IF (M1.NE.2) GO TO 80
      XD(2)=XD(1)+DEL/3.
```

```
    XD(3)=XD(2)+DEL/3.
80  IF (M2.NE.1) GO TO 90
    XD(3+M1)=XD(2+M1)+DEL/2.
90  IF((NCLL.NE.2).AND.(NCRR.NE.2))GO TO 50
    CALL UGLYDK(ND,NCLL,NCRR,XD,SCP,ESL,ESR,AS,ISP)
    CALL EVALDK(ND,NP,XD,XP,SCP,AS,ISP)
    DO 60 I=2,NP1
60  YSP(I)=YSP(I)-SCP(I)
50  CONTINUE
    NCL=1
    DO 2 N=1,ND2
    DO 3 L=1,ND
 3  YD(L)=0.0
    YD(N+1)=1.0
    CALL UGLYDK(ND,NCLL,NCRR,XD,YD,0.0,0.0,AE(1,N),ITS)
    CALL EVALDK(ND,NF,XD,XF,YQ(NCL),AE(1,N),ITS)
 2  NCL=NCL+NF
 10 IF(NF.EQ.NP) GO TO 11
    CALL UGLYDK(NP,NCLL,NCRR,XP,YSP,0.0,0.0,AG,ITS)
    CALL EVALDK(NP,NF,XP,XF,YF,AG,ITS)
    GO TO 12
 11 DO 13 N=1,NP
 13 YF(N)=YSP(N)
 12 DO 4 N=1,NF
 4  YY(N)=YF(N)
    DO 14 N=1,NMX
 14 AA(N)=YQ(N)
    CALL LLSQ(AA,YY,NF,ND2,1,C,IPIV,0.00001,IER,AUX)
    IF(IER.NE.0) RETURN
    NM1=ND-1
```

```
      DO 5 N=1,NM1
      N1=4*(N-1)+1
      N2=N1+1
      N3=N2+1
      N4=N3+1
      AF(N1)=AS(N1)
      AF(N2)=AS(N2)
      AF(N3)=SLP+AS(N3)
      AF(N4)=YBG+SLP*(XD(N)-XBG)+AS(N4)
      DO 5 J=1,ND2
      AF(N1)=AF(N1)+C(J)*AE(N1,J)
      AF(N2)=AF(N2)+C(J)*AE(N2,J)
      AF(N3)=AF(N3)+C(J)*AE(N3,J)
5      AF(N4)=AF(N4)+C(J)*AE(N4,J)
      RETURN
      END
```

Appendix I

```
TEST OF PROGRAM WITH model666
50  15  60  1  7.  2.0  0  10.          kn.nt,nx,kode,h,rho,npitch,pitch
1  1.                                          icode, excentricity (a/b)
1.0 0.0  0.0                          vx vy vz
12.4  0.0  0.0                          center of gravity
4 1 0.0 0.01583 0.065 0.145833 0.2583 0.3992 0.5675 0.7625 0.9808 1.083 1.1042
1.2117 1.225 1.4833 1.7667 2.0583 2.375 2.7 3.0333 3.375 3.725 4.075 4.4417
4.7917 5.15 7.6667 10.1667 11.8667 14.45 17 17.5 18 18.5 19 19.5 20 20.5 20.97
 21.5 22 22.5 23 23.5 24 24.5 25 25.5 26.0 26.5 27.0
0.0 0.324253 0.546149 0.739448 0.913357 1.072906 1.219077 1.355185 1.479878
1.52897 1.53928 1.588372 1.594508 1.698829 1.794926 1.880837 1.957544 2.025045
 2.083342 2.132434 2.173303 2.20509 2.228531 2.243873 2.250009 2.250009
2.250009 2.250009 2.250009 2.250009 2.244977 2.22755 2.199935 2.160048 2.109974
 2.047505 1.974971 1.890042 1.794926 1.687538 1.569962 1.439991 1.299956
1.147526 0.984908 0.810018 0.624941 0.427469 0.219932 0.0 0.0 0.0
```



S075-31