Appendix A

Vorticity Conservation Conditions

This appendix contains a derivation of the vorticity conservation conditions for incompressible flow with uniform density and kinematic viscosity. This derivation is based on the outline by Wu in Ref. [45] and [72] and is derived here for the reader’s convenience. Begin with the continuity equation for an incompressible flow, the Navier-Stokes equation, and the definition of vorticity which are shown below in a respective order.

\[ \nabla \cdot \mathbf{V} = 0 \quad \text{(A.1)} \]

\[ \frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V} \cdot \nabla) \mathbf{V} = -\frac{1}{\rho} \nabla P + \nu \nabla^2 \mathbf{V} \quad \text{(A.2)} \]

\[ \nabla \times \mathbf{V} = \Omega \quad \text{(A.3)} \]

This derivation involves several vector identities which are shown in Eq. A.4, A.5, A.6, A.7, and A.8 [73].

\[ \nabla^2 \mathbf{V} = \nabla(\nabla \cdot \mathbf{V}) - \nabla \times (\nabla \times \mathbf{V}) \quad \text{(A.4)} \]

\[ \nabla \times (\nabla P) = 0 \quad \text{(A.5)} \]

\[ \nabla \times (\mathbf{V} \times \Omega) = \mathbf{V}(\nabla \cdot \Omega) - \Omega(\nabla \cdot \mathbf{V}) + (\Omega \cdot \nabla) \mathbf{V} - (\mathbf{V} \cdot \nabla) \Omega \quad \text{(A.6)} \]
APPENDIX A. VORTICITY CONSERVATION CONDITIONS

\[(\mathbf{V} \cdot \nabla)\mathbf{V} = \frac{1}{2} \nabla (\mathbf{V} \cdot \mathbf{V}) - \mathbf{V} \times (\nabla \times \mathbf{V})
\]

\[= \frac{1}{2} \nabla (\mathbf{V} \cdot \mathbf{V}) - \mathbf{V} \times \Omega \quad (A.7)\]

\[\nabla \cdot (\nabla \times \mathbf{V}) = 0 \quad (A.8)\]

Equation A.4 is a vector identity for the Laplacian of a vector and is written here for the velocity vector. Equation A.5 states that the curl of the gradient of a scalar is always zero. This identity is written here for the scalar, pressure, although it is valid for any scalar and will be used for other scalars in this derivation. Equation A.6 is a vector identity involving any two vectors and is written here in the form it will be used involving the velocity and vorticity vectors. Equation A.7 is a vector identity for a single vector and is shown here for velocity along with the definition for vorticity. Equation A.8 is a vector identity which states that the divergence of the curl of a vector is always zero. Written here for the velocity vector, we see that vorticity is a solenoidal vector field.

Since all of the vorticity conservation conditions stem from the vorticity transport equation, the derivation of the latter will be the first step. If we apply the vector identity of Eq. A.4 along with the definition of vorticity in Eq. A.3 and the continuity equation in Eq. A.1 to replace the diffusion term in the Navier-Stokes equation (Eq. A.2), the result is Eq. A.9.

\[\frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V} \cdot \nabla)\mathbf{V} = -\frac{1}{\rho} \nabla P - \nu \nabla \times \Omega \quad (A.9)\]

The next step towards obtaining the vorticity transport equation is to take the curl of this equation. This can be done by taking the curl of each term individually as is shown in Eq. A.10.

\[\nabla \times \frac{\partial \mathbf{V}}{\partial t} + \nabla \times [(\mathbf{V} \cdot \nabla)\mathbf{V}] = \nabla \times \left[-\frac{1}{\rho} \nabla P\right] - \nabla \times [\nu \nabla \times \Omega] \quad (A.10)\]

For the first term on the left hand side of Eq. A.10, the curl can be brought inside the partial derivative with respect to time and along with the definition of vorticity
can be written as the partial derivative of vorticity with respect to time. The first term on the right hand side can be set to zero due to the vector identity of Eq. A.5. Using the vector identity of Eq. A.4 for the vorticity vector in the second term on the right hand side we can rewrite Eq. A.10 as is shown in Eq. A.11.

\[
\frac{\partial \Omega}{\partial t} + \nabla \times [(V \cdot \nabla)V] = \nu \nabla^2 \Omega \tag{A.11}
\]

Using the vector identity of Eq. A.7, the second term on the left hand side of Eq. A.11 can be replaced

\[
\nabla \times [(V \cdot \nabla)V] = -\nabla \times (V \times \Omega) \tag{A.12}
\]

resulting in

\[
\frac{\partial \Omega}{\partial t} = \nabla \times (V \times \Omega) + \nu \nabla^2 \Omega \tag{A.13}
\]

Using the vector identities of Eq. A.6 and A.8 along with the continuity equation, the first term on the right hand side of Eq. A.13 can be rewritten and Eq. A.14, the vorticity transport equation for a 3-D, incompressible, uniform density, uniform viscosity flow is obtained.

\[
\frac{\partial \Omega}{\partial t} + (V \cdot \nabla)\Omega = (\Omega \cdot \nabla)V + \nu \nabla^2 \Omega
\]

\[
\frac{D\Omega}{Dt} = (\Omega \cdot \nabla)V + \nu \nabla^2 \Omega \tag{A.14}
\]

This equation closely resembles the form of the Navier-Stokes equation with the exception of one additional term which has no counterpart. The first term on the right hand side of Eq. A.14 describes the change in vorticity due to the stretching of a vortex line. It should also be noted here that the description of a flow by vorticity conservation is done without having to deal with a pressure term. This is seen as a major advantage over the corresponding momentum conservation. This equation can be written in a slightly different manner by applying the vector identity of Eq. A.4 written for vorticity along with the vector identity of Eq. A.8 on the last term on the right hand side of the vorticity transport equation. This is shown in Eq. A.15, which is the starting point for the derivation of the vorticity conservation conditions for several physical situations.
\[ \frac{D\Omega}{Dt} = (\Omega \cdot \nabla) V - \nu \nabla \times (\nabla \times \Omega) \quad (A.15) \]

Before the derivation is continued separately for 2-D and 3-D flow it is useful to define the physical regions and boundaries of the flowfields in question. For 3-D flow, a region refers to a volume and a surface refers to that which forms the boundary to the region. Two-dimensional flow is an infinitely long extrusion along a single coordinate direction, so for these flows, a region refers to the in-plane area and the surface refers to the closed curve that forms the boundary to that region. The entire limitless region occupied by both solids and fluids is referred to as \( R_\infty \). This region is only bounded externally by a surface at infinity, \( S_\infty \). Solid regions are referred to as \( R_j \) where the subscript \( j \) is the index number of the solid object. These regions are externally bounded by surfaces, \( S_j \), with corresponding indices \( j \). The fluid region, \( R_f \) is externally bounded by \( S_\infty \) and internally bounded by \( S_S \), where \( S_S \) signifies all of the solid surfaces collectively. Normal vectors point to the exterior of their associated region. A schematic diagram of the different regions and surfaces can be seen in Fig. A.1.

For 2-D flow, the vorticity and velocity are always orthogonal to one another with the velocity vector in the plane of the flow and the vorticity vector in the direction normal to the plane of flow. Therefore, the vorticity is also orthogonal to the gradient of velocity, and Eq. A.15 reduces to Eq. A.16.

\[ \frac{D\Omega}{Dt} = -\nu \nabla \times (\nabla \times \Omega) \quad (A.16) \]

It should also be noted here that the vorticity vector does not have a component normal to a solid fluid boundary. Equation A.17 is a relation between surface and volume integrals and is written here for any arbitrary vector, \( q \).

\[ \int \int \int_R \nabla \times q dR = -\oint_S q \times n dS \quad (A.17) \]

If we integrate Eq. A.16 over the fluid region, \( R_f \), and apply Eq. A.17 to the right hand side we obtain Eq. A.18.

\[ \frac{d}{dt} \int_{R_f(t)} \Omega dR = \nu \int_{S_S(t)} (\nabla \times \Omega) \times n dS \quad (A.18) \]
The surface integral is written here solely over the solid surface, since there can be no diffusion at the infinity boundary. The left hand side results from applying Reynolds transport theorem along with the continuity equation to the integral of the total derivative of vorticity. The right hand side of Eq. A.18 can now be replaced with the use of the Navier-Stokes equation. If we take the cross product of each term in Eq. A.9 with the solid surface normal vector and integrate over the solid surface, we obtain Eq. A.19, where the normal vector points to the exterior of the fluid region, \( R_f \).

\[
\int_{S_s} \frac{DV}{Dt} \times n dS = -\frac{1}{\rho} \int_{S_s} \nabla \times P \times n dS - \nu \int_{S_s} (\nabla \times \Omega) \times n dS
\]  

(A.19)

The first term on the right hand side is always zero. This can easily be seen by transforming the term to a volume integral using Eq. A.17. The integrand, \( \nabla \times \nabla P \), is always zero as stated by the vector identity in Eq. A.5. Replacing the right hand side of Eq. A.18 with the left hand side of Eq. A.19 results in Eq. A.20.

\[
\frac{d}{dt} \int_{R_f(t)} \Omega dR = - \int_{S_s} \frac{DV}{Dt} \times n dS
\]  

(A.20)

We now make use of the no-slip boundary condition on the solid surface which simply states that at the boundary \( S_s \) the total derivative of velocity with respect to time is identical for both the fluid and solid. Therefore, a version of Eq. A.20 can be written for the entire region, \( R_\infty \). This can be seen in Eq. A.21 where the right hand side comes from applying Eq. A.17 to the right hand side of Eq. A.20 with the unit normal vector pointing to the exterior of the solid region instead of to the exterior of the fluid region.

\[
\frac{d}{dt} \int_{R_f(t)} \Omega dR = - \sum_j \int_{R_j} \nabla \times \frac{DV}{Dt} dR \\
\frac{d}{dt} \int_{R_f(t)} \Omega dR = - \sum_j \int_{R_j} \frac{D\Omega}{Dt} dR \\
\frac{d}{dt} \int_{R_f(t)} \Omega dR = - \sum_j \frac{d}{dt} \int_{R_j} \Omega dR
\]  

(A.21)

This results in Eq. A.22.

\[
\frac{d}{dt} \int_{R_f} \Omega dR + \sum_j \frac{d}{dt} \int_{R_j} \Omega dR = 0
\]
\[ R_f + \sum_j R_j = R_\infty \]
\[ \frac{d}{dt} \iiint_{R_\infty} \Omega dR = 0 \]  \hspace{1cm} (A.22)

So, for 2-D, incompressible, uniform density, uniform viscosity flow, the total vorticity for the combined region (fluid and solid) is constant with time. The vorticity of the solid region is simply twice its angular velocity.

For 2-D, inviscid flow it can easily be seen from the 2-D vorticity transport equation (Eq. A.16) that the diffusion term on the right hand side is equal to zero. The resulting equation is the 2-D version of Kelvin’s Theorem, shown here in Eq. A.23.

\[ \frac{D\Omega}{Dt} = 0 \]  \hspace{1cm} (A.23)

It states that the vorticity of a fluid element remains constant with time. This work is stated here for an incompressible fluid although it can be shown that this holds for the more general case of a barotropic fluid regardless of compressibility [47]. For consistency, the vorticity conservation condition for the inviscid case, which is presented as a stipulation on the substantial derivative of vorticity, can be rewritten in an integral form. This differential equation holds for the fluid region, so if we integrate Eq. A.23 over the region \( R_f \) and apply Reynolds transport theorem we obtain the following equation.

\[ \frac{d}{dt} \iiint_{R_f} \Omega dR = 0 \]  \hspace{1cm} (A.24)

This equation closely resembles the vorticity conservation condition for 2-D viscous flow in Eq. A.22 except the integral excludes the solid regions. This is a result of the absence of a no-slip condition, so vorticity in the fluid region has no relation to the vorticity in the solid region which is equal to two times the angular velocity of a solid body rotation. If the solid bodies have zero angular velocity, Eq. A.22 is identical to the corresponding inviscid condition in Eq. A.24.

For 3-D flow the approach to deriving the vorticity conservation expression differs greatly from the approach involving the integration of Eq. A.15 over the regions. This is due to the fact that in order to maintain complete generality of the flow situation
the \((\mathbf{\Omega} \cdot \nabla) \mathbf{V}\) term cannot be eliminated. The vorticity vector has a component normal to the solid fluid boundaries and due to the no-slip condition this normal component is continuous across that boundary. The derivation makes use of two vector identities shown in Eq. A.25 and A.26 where \(\psi\) and \(\phi\) are scalars or vectors as indicated by bold face type.

\[
\nabla \cdot (\psi \mathbf{\phi}) = \psi \nabla \cdot \mathbf{\phi} + \mathbf{\phi} \cdot \nabla \psi \quad (A.25)
\]

\[
\nabla(\psi \cdot \mathbf{\phi}) = (\nabla \mathbf{\phi}) \psi + (\nabla \psi) \mathbf{\phi} \quad (A.26)
\]

These vector identities can be used to derive Eq. A.28 with the manipulations shown in Eq. A.27. This analysis also notes that vorticity is solenoidal and that we are assuming that \(\mathbf{a}\) is a constant vector. The gradient of the position vector, \(\mathbf{r}\), is the identity matrix.

\[
\nabla \cdot [(\mathbf{a} \cdot \mathbf{r}) \mathbf{\Omega}] = (\mathbf{a} \cdot \mathbf{r}) (\nabla \cdot \mathbf{\Omega}) + \mathbf{\Omega} \cdot \nabla (\mathbf{a} \cdot \mathbf{r})
\]

\[
\nabla (\mathbf{a} \cdot \mathbf{r}) = (\nabla \mathbf{a}) \mathbf{r} + (\mathbf{r} \nabla) \mathbf{a} \quad (A.27)
\]

\[
\nabla \cdot [(\mathbf{a} \cdot \mathbf{r}) \mathbf{\Omega}] = \mathbf{a} \cdot \mathbf{\Omega} \quad (A.28)
\]

The derivation will also use the Divergence Theorem which is shown for an arbitrary vector, \(\mathbf{q}\), in Eq. A.29.

\[
\iiint_{R} \nabla \cdot \mathbf{q} dR = \oiint_{S} \mathbf{q} \cdot \mathbf{n} dS \quad (A.29)
\]

If the integrand in the volume integral of the Divergence Theorem is replaced with the left hand side of Eq. A.28 and the constant vector, \(\mathbf{a}\), is cancelled out on both sides of the equation, the resulting equation is Eq. A.30.

\[
\iiint_{R_f} \mathbf{\Omega} dR = \oiint_{S_f} \mathbf{r} (\mathbf{\Omega} \cdot \mathbf{n}) dS \quad (A.30)
\]

This equation is written for the fluid region, \(R_f\). A similar equation can be written for the solid region as in Eq. A.31 taking note that the normal vector for this equation is pointing opposite to that for the equations written for the fluid region.

\[
\iiint_{R_s} \mathbf{\Omega} dR = \oiint_{S_s} \mathbf{r} (\mathbf{\Omega} \cdot \mathbf{n}) dS \quad (A.31)
\]
Due to the no-slip condition, the right hand side of Eq. A.31 is the negative of the right hand side of Eq. A.30, and we can write

\[ \iiint_{R_f} \mathbf{\Omega} d\mathbf{R} = - \iiint_{R_S} \mathbf{\Omega} d\mathbf{R} \]  

(A.32)

Therefore, for the 3-D, incompressible, uniform density, uniform viscosity flow, the vorticity conservation statement is that the integral of the vorticity over the combined fluid and solid regions is always zero.

\[ \iiint_{R_{\infty}} \mathbf{\Omega} d\mathbf{R} = 0 \]

\[ R_{\infty} = R_f + R_s \]  

(A.33)

For 3-D, inviscid flow, it can easily be seen from the 3-D vorticity transport equation (Eq. A.15) that the diffusion term on the right hand side is equal to zero leaving

\[ \frac{D\mathbf{\Omega}}{Dt} = (\mathbf{\Omega} \cdot \nabla)\mathbf{V} \]  

(A.34)

This is the 3-D version of Kelvin’s theorem which is identical to the 2-D equation in Eq. A.23 if the flow starts from rest, i.e. no initial vorticity. Similar to our treatment of the 2-D Kelvins theorem, we can integrate Eq. A.34 over the region \( R_f \) and apply Reynolds transport theorem to obtain

\[ \iiint_{R_f} \mathbf{\Omega} d\mathbf{R} = \oiint_{S_S} \mathbf{r} (\mathbf{\Omega} \cdot \mathbf{n}) dS \]  

(A.35)

According to the Helmholtz Vorticity Theorems, a vortex filament in an inviscid flow cannot end at a solid boundary, so the right hand side of Eq. A.35 is zero resulting in Eq. A.36.

\[ \iiint_{R_f} \mathbf{\Omega} d\mathbf{R} = 0 \]  

(A.36)

In review, the four vorticity conservation equations derived here can be found in Eq. A.22, A.33, A.23, and A.34 and are presented here again for easy reference.

\[
\begin{cases}
3 - D \\
2 - D
\end{cases}
\begin{align*}
\text{Viscous:} & \quad \iiint_{R_{\infty}} \mathbf{\Omega} d\mathbf{R} = 0 \\
\text{Inviscid:} & \quad \iiint_{R_f} \mathbf{\Omega} d\mathbf{R} = 0 \\
\text{Viscous:} & \quad \frac{d}{dt} \oiint_{R_{\infty}} \mathbf{\Omega} dS = 0 \\
\text{Inviscid:} & \quad \frac{d}{dt} \oiint_{R_f} \mathbf{\Omega} dS = 0
\end{align*}
\]  

(A.37)
Figure A.1: Regions and Boundaries in Flowfield
Appendix B

2-D Vortex Panel Method Solution

The flow solution is obtained as a linear superposition of the continuous vortex sheet, point vortices, and uniform free stream flow, all of which are solutions to Laplace's equation, in such a way as to satisfy the no-penetration boundary condition of the Neumann type. The continuous vortex sheet is comprised of $N$ panels with piecewise linear vortex sheet strength connected at $N + 1$ nodes. The boundary condition is imposed at a single collocation point located at the midpoint of each panel. The boundary condition is shown in Eq. B.1

$$ (V_B + V_w + V_\infty) \cdot n = 0 \quad (B.1) $$

$V_B$ is the velocity induced by the boundary layer, $V_w$, is the velocity induced by the shed wake, and $V_\infty$ is the velocity induced by the free stream flow. The vorticity conservation condition is simultaneously imposed.

$$ \frac{1}{2} \sum_{i=1}^{N} (\gamma_i + \gamma_{i+1}) \Delta l_i + \Gamma_t + \sum_{k=1}^{M} \Gamma_k = \text{Constant} \quad (B.2) $$

Throughout this work, $i$ is the index for the surface panel inducing a velocity, $k$ is the index for the wake point vortices inducing a velocity, and $j$ is the index for the panel at which the no-penetration boundary condition is being imposed. The velocity induced by the vortex sheet, $V_B$, is composed of the $N$ linear sections of the sheet plus a point vortex at the trailing edge as indicated in Eq. B.3. This point vortex
replaces the vortex sheet at the trailing edge and easily allows for the toggling between conditions for the Giesing/Maskell trailing edge model.

\[
V_B(r, t) = \sum_{i=1}^{N} V_{b_i}(r, t) - \frac{1}{2\pi} e_z \times \frac{\Gamma_t(t)(r - r_t)}{|r - r_t|^2} \tag{B.3}
\]

Equation B.4 describes the velocity induced by the shed wake as the summation of the velocities induced by the \( M \) individual point vortices that comprise the wake.

\[
V_w(r, t) = -\frac{1}{2\pi} e_z \times \sum_{k=1}^{M} \Gamma_k \frac{r - r_k}{|r - r_k|^2 + \sigma^2} \tag{B.4}
\]

The uniform free stream is taken as magnitude unity and is shown in Eq. B.5 where \( \alpha \) is the angle of attack.

\[
V_{\infty} = \cos \alpha E_x + \sin \alpha E_y \tag{B.5}
\]

As was previously mentioned, the body is discretized into \( N \) panels. The solution takes place in two reference frames; a global reference frame and a panel local reference frame. The discretized body, along with the two reference frames are shown in Fig. B.1 for a 2-D airfoil case. Upper case letters refer to values in the global coordinate system, while lower case letters refer to values in the panel local coordinate system. The boundary condition is imposed in the global coordinate system. Equation B.6 shows the length of panel \( i \)

\[
\Delta l_i = \sqrt{(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2} \tag{B.6}
\]

and Eq. B.7 shows the direction cosines for the coordinate transformation between global and panel local coordinate systems.

\[
d_1 = \frac{(X_{i+1} - X_i)}{\Delta l_i} \quad d_2 = \frac{(Y_{i+1} - Y_i)}{\Delta l_i} \tag{B.7}
\]

The transformation from global to panel local coordinate system can be seen in Eq. B.8.

\[
x = d_1(X_j - X_i) + d_2(Y_j - Y_i) \\
y = -d_2(X_j - X_i) + d_1(Y_j - Y_i) \tag{B.8}
\]
Equations B.9 through B.12 show the ingredients for the calculation of the contribution of the linearly varying vortex sheet on panel $i$ to the velocity (Eq. B.13).

\[
\Delta \theta = \tan^{-1}\left(\frac{x}{y}\right) - \tan^{-1}\left(\frac{x - \Delta l_i}{y}\right) \tag{B.9}
\]

\[
R = \frac{(x - \Delta l_i)^2 + y^2}{x^2 + y^2} \tag{B.10}
\]

\[
v_{1xi} = \frac{1}{2\pi \Delta l_i}[(\Delta l_i - x)\Delta \theta - \frac{1}{2}y \ln R]
\]

\[
v_{1yi} = \frac{1}{2\pi \Delta l_i}[y \Delta \theta - \Delta l_i + \frac{1}{2}(\Delta l_i - x) \ln R]
\]

\[
v_{2xi} = \frac{1}{2\pi \Delta l_i - i}[x \Delta \theta + \frac{1}{2}y \ln R]
\]

\[
v_{2yi} = \frac{1}{2\pi \Delta l_i}[\Delta l_i - y \Delta \theta + \frac{1}{2}x \ln R] \tag{B.11}
\]

\[
V_{1Xi} = d_1 v_{1xi} - d_2 v_{1yi}
\]

\[
V_{1Yi} = d_2 v_{1xi} + d_1 v_{1yi}
\]

\[
V_{2Xi} = d_1 v_{2xi} - d_2 v_{2yi}
\]

\[
V_{2Yi} = d_2 v_{2xi} + d_1 v_{2yi} \tag{B.12}
\]

\[
V_b(X, Y, t) = \gamma_i [V_{1Xi} E_X + V_{1Yi} E_Y] + \gamma_{i+1} [V_{2Xi} E_X + V_{2Yi} E_Y] \tag{B.13}
\]

The velocity induced by the infinitely thin boundary layer from Eq. B.3 can be seen in terms of the global coordinates in Eq. B.14.

\[
V_B = \sum_{i=1}^{N} V_{bi} + \frac{1}{2\pi} \Gamma(t) \frac{(Y - Y_t) E_X - (X - X_t) E_Y}{(X - X_t)^2 + (Y - Y_t)^2} \tag{B.14}
\]

Likewise, the velocity induced by the wake and free stream can be seen in Eq. B.15 and B.16 respectively in terms of the global coordinates.

\[
V_w(X, Y, t) = \frac{1}{2\pi} \sum_{k=1}^{M} \Gamma_k \frac{(Y - Y_k) E_X - (X - X_k) b f E_Y}{(X - X_k)^2 + (Y - Y_k)^2 + \sigma^2} \tag{B.15}
\]

\[
V_\infty(X, Y, t) = \cos \alpha E_X + \sin \alpha E_Y \tag{B.16}
\]
At each time step, the Giesing/Maskell trailing edge model is used to determine the value of $\Gamma_k$ which is the $\Delta \Gamma_w$ for that time step using Eq. B.17.

$$
\begin{align*}
\Gamma_t > 0 & \quad \left\{ \begin{array}{l}
\Gamma_t = \frac{\gamma_w \Delta U}{2} \\
\Delta \Gamma_w = \frac{2\Gamma_t^2}{\Delta l} \Delta t
\end{array} \right.
\Gamma_t < 0 & \quad \left\{ \begin{array}{l}
\Gamma_t = \frac{\gamma_l \Delta l}{2} \\
\Delta \Gamma_w = -\frac{2\Gamma_t^2}{\Delta l} \Delta t
\end{array} \right.
\end{align*}
$$

(B.17)

Equation B.18 shows the boundary condition with the unit normal written out explicitly in terms of the global coordinates.

$$(V_B + V_w + V_\infty) \cdot n = 0$$

$$(V_B + V_w + V_\infty) \cdot \left[ -\frac{(Y_{j+1} - Y_j)}{\Delta l_j} E_X + \frac{(X_{j+1} - X_j)}{\Delta l_j} E_Y \right] = 0 \quad \text{(B.18)}$$

The dot product of each velocity contributor with the unit outward normal of a panel $j$ on which the boundary condition is being imposed can be seen in Eq. B.19, B.20, and B.21 for the vortex sheet, wake, and free stream respectively.

$$
V_B \cdot n = \sum_{i=1}^{N} (-\gamma_i V_{1Xi} + \gamma_{i+1} V_{2Xi}) \left( \frac{Y_{j+1} - Y_j}{\Delta l_j} \right) + (\gamma_i V_{1Yi} + \gamma_{i+1} V_{2Yi}) \left( \frac{X_{j+1} - X_j}{\Delta l_j} \right)
$$

$$
-\frac{1}{2\pi} \Gamma_t(t) \left( \frac{Y_j - Y_t}{(X_j - X_t)^2 + (Y_j - Y_t)^2} \frac{Y_{j+1} - Y_j}{\Delta l_j} \right)
$$

$$
-\frac{1}{2\pi} \Gamma_t(t) \left( \frac{X_j - X_t}{(X_j - X_t)^2 + (Y_j - Y_t)^2} \frac{X_{j+1} - X_j}{\Delta l_j} \right)
$$

(B.19)

$$
V_w \cdot n = \frac{1}{2\pi} \sum_{k=1}^{M} \Gamma_k \left[ -\frac{(Y_j - Y_k)}{(X_j - X_k)^2 + (Y_j - Y_k)^2 + \sigma^2} \left( \frac{Y_{j+1} - Y_j}{\Delta l_j} \right) \right]
$$

$$
-\frac{(X_j - X_k)}{(X_j - X_k)^2 + (Y_j - Y_k)^2 + \sigma^2} \left( \frac{X_{j+1} - X_j}{\Delta l_j} \right)
$$

(B.20)

$$
V_\infty \cdot n = -\cos \alpha \left( \frac{Y_{j+1} - Y_j}{\Delta l_j} \right) + \sin \alpha \left( \frac{X_{j+1} - X_j}{\Delta l_j} \right)
$$

(B.21)
The boundary condition, Eq. B.1 can be separated into the part that needs to be solved for and the part that is known from previous time steps (right hand side) as shown in Eq. B.22.

\[
V_B \cdot n = -(V_w + V_\infty) \cdot n
\]  

At each time step, another point vortex is shed and added to the wake. Each point vortex is also convected at the local velocity, so that the new point vortex location is equal to the old location plus the product of the velocity and the time step. So at each time step, the right hand side (RHS) is updated and the unknown strengths of the vortex sheet at the panel nodes along with the trailing edge point vortex are calculated by solving the linear system of equations (Eq. B.23), where \(a_{j,i}\) is given in Eq. B.24.

\[
\begin{bmatrix}
  a_{j,i} \\
  \vdots \\
  \gamma_n \\
  \Gamma_t
\end{bmatrix}
= 
\begin{bmatrix}
  \text{RHS}
\end{bmatrix}
\]  

(B.23)

\[
a_{j,i} = -V_2 X_{i-1} \left( \frac{Y_{j+1} - Y_j}{\Delta l_j} \right) + V_2 Y_{i-1} \left( \frac{X_{j+1} - X_j}{\Delta l_j} \right) \\
- V_1 X_i \left( \frac{Y_{j+1} - Y_j}{\Delta l_j} \right) + V_1 Y_i \left( \frac{X_{j+1} - X_j}{\Delta l_j} \right)
\]  

(B.24)

Since the trailing edge flow is described using the point vortex of strength \(\Gamma_t\) the vortex sheet strength is set to zero at the trailing edge (\(\gamma_1 = \gamma_{n+1} = 0\)), therefore they are excluded from the system of equations. It can be seen that the coefficient for the \(\gamma\) at the node involves a \(V_2\) from the panel behind the node \((i - 1)\) and a \(V_1\) from the panel ahead of the node \((i)\). Figure B.2 shows two adjacent panels and their shared node along with the linear hat function for the velocity distribution over the two panels associated with the sheet strength at the node. It should be noted that the \(V\) functions are not the linear hat functions themselves but the influence coefficients of the linear sheet strength distribution on panel \(j\). The matrix entries for the vorticity conservation equation are shown in Eq. B.25.

\[
a(N + 1, i) = \frac{1}{2} \Delta l_{i-1} + \frac{1}{2} \Delta l_i
\]  

(B.25)
APPENDIX B. 2-D VORTEX PANEL METHOD SOLUTION

Caution must be taken in the calculation of the diagonal elements of the $a$ matrix, $a_{i,i}$, since the values of $y$ are zero. When calculating $\Delta \theta$ it must be explicitly specified from which side of the panel zero is being approached. To avoid this problem, the values for this special case are explicitly provided in Eq. B.26 instead of being calculated using Eq. B.11.

\[
\begin{align*}
  x &= \frac{\Delta l}{2} \quad \Delta \theta = -\pi \\
  y \to 0^+ &\quad R = 1 \\
  v_{1x_i} &= -\frac{1}{4} \\
  v_{1y_i} &= -\frac{1}{2\pi} \\
  v_{2x_i} &= -\frac{1}{4} \\
  v_{2y_i} &= \frac{1}{2\pi}
\end{align*}
\]  

(B.26)

The same approach is taken for the solution of the bluff body problem. For this case, there are two separation points and, therefore, two wakes. The bluff body extension to the model does not employ point vortices on the surface of the solid region, such as the one placed at the trailing edge of the streamlined body so the vector of unknowns consists only of the vortex sheet strengths at the $N$ distinct nodes (Eq. B.27).

\[
\begin{bmatrix}
  a_{j,i} \\
  \vdots \\
  a_{j,i}
\end{bmatrix}
\begin{bmatrix}
  \gamma_1 \\
  \vdots \\
  \gamma_n
\end{bmatrix}
= 
\begin{bmatrix}
  RHS
\end{bmatrix}
\]  

(B.27)
Panel i is bounded by nodes i and i+1

Figure B.1: Discretized Geometry and Coordinate Systems
Figure B.2: Adjacent Panels and Hat Functions
Appendix C

2-D Turbulent Gap Flow Calculation

The fully-developed, turbulent flow in a thin 2-D gap with one wall moving can be analyzed by solving Eq. C.1, which is the momentum equation for such a flow.

\[
\frac{dp}{dx} = \frac{d}{dy} \left[ (\mu + \mu_T) \frac{du}{dy} \right]
\]

\[
u_0 = 0.0
\]

\[
u_h = u_\infty
\]

This simply states that the pressure gradient in the longitudinal direction of the pipe is equal to the shear stress gradient in the lateral direction of the pipe. The boundary condition at the two pipe walls forces a zero velocity at the lower wall and a possibly none zero velocity at the upper wall. A schematic diagram of the flow can be seen in Fig. C.1. Equation C.2 results from integrating both sides of Eq. C.1 with respect to \( y \).

\[
\frac{dp}{dx} y = (\mu + \mu_T) \frac{du}{dy} - \mu \frac{du}{dy} \bigg|_0
\]

Integrating a second time with respect to \( y \) yields

\[
u - u_0 = \int_0^y \frac{dp}{dx} y' dy' + \int_0^y \frac{du}{\mu + \mu_T} dy'
\]
APPENDIX C. 2-D TURBULENT GAP FLOW CALCULATION

The boundary condition at \( y = 0 \), \( u(0) = 0 \), allows us to eliminate \( u_0 \) and the boundary condition at \( y = h \), \( u(h) = u_\infty \), allows us to solve for the velocity gradient at the \( y = 0 \) wall.

\[
\frac{du}{dy}igr|_0 = \frac{u_\infty - \int_0^h \frac{\partial u'}{\partial y'} dy'}{\int_0^h \frac{\mu + \mu_T}{\mu + \mu_T} dy'}
\]  \hfill (C.4)

If Eq. C.4 is inserted into Eq. C.3 we obtain the solution for longitudinal velocity at any height, \( y \).

\[
u(y) = \int_0^y \frac{\partial p}{\partial x} dy' + \frac{u_\infty - \int_0^h \frac{\partial u'}{\partial y'} dy'}{\int_0^h \frac{\mu}{\mu + \mu_T} dy'} \int_0^y \frac{\mu}{\mu + \mu_T} dy' \]  \hfill (C.5)

The eddy viscosity is modeled separately in the inner and outer regions. These models are the Reichart turbulence model and a pipe flow outer region model from Reynolds shown in Eq. C.6 in a respective order.

\[
\begin{align*}
\mu_T &= kp\nu \left[ \left( \frac{y_+}{\nu} \right) - y_+^+ \tanh \left( \frac{y_+}{\nu y_+^+} \right) \right] \\
\mu_T &= 0.192kp\nu h
\end{align*}
\]  \hfill (C.6)

The friction velocity, \( u_+ \), is dependent on the final solution so the integration, Eq. C.5, must be solved iteratively within a root finding scheme to update values of the wall shear stress.
Figure C.1: 2-D Pipe Flow with a Moving Wall and Pressure Gradient
Appendix D

Green’s Identity Formulation

For this outline of the Green’s Identity formulation of a linear aerodynamic analysis, i.e. a panel method, one is looking at the exterior flow in a region with only one interior boundary. The extension for multiple interior boundaries is trivial. The flow is divided into two regions, \( R_f \), the region of interest, and \( R_i \). The exterior boundary of \( R_f \) is \( S_\infty \) and the interior boundary is \( S \) which is composed of \( S_B \), the surface of solid body, and \( S_w \), the surface of a shed wake. The normal vector of the boundaries point to the outside of the region being considered. These regions can be seen pictorially in Fig. D.1. Laplace’s equation must hold in both the interior region, \( R_i \) and the exterior region, \( R_f \),

\[
\nabla^2 \Phi = 0 \\
\n\nabla^2 \Phi_i = 0
\]

where \( \Phi \) is the total velocity potential which is equal to a free stream potential plus a perturbation potential, \( \phi \).

\[
\Phi = \phi_\infty + \phi
\]

Green’s Theorem is used to find the velocity potential at a point, \( P \). Green’s Theorem is the Divergence Theorem written for a vector composed of two scalar functions of position, \( \Phi_1 \nabla \Phi_2 - \Phi_2 \nabla \Phi_1 \).

\[
\iint_S (\Phi_1 \nabla \Phi_2 - \Phi_2 \nabla \Phi_1) \cdot \mathbf{n} dS = \iiint_R (\Phi_1 \nabla^2 \Phi_2 - \Phi_2 \nabla^2 \Phi_1) dR
\]
If the scalars of Green’s Theorem are solutions to Laplace’s equation, the right hand side of Eq. D.3 equals zero. These harmonic functions are placed on the interior boundary, and the no-penetration condition (Dirichlet) is imposed on the boundaries. The formulation is slightly different for 2-D and 3-D flows, so the discussion is continued in separate sections for the two cases. Detailed discussions can be found in Refs. [47], [48], and [52].

### D.1 2-D Flows

For the 2-D formulation, set \( \Phi_1 = \ln r \) and \( \Phi_2 = \Phi \) where \( r \) is the scalar distance from some point, \( P \). \( \Phi_1 \) and \( \Phi_2 \) are both solutions to Laplace’s equation, so the right hand side of Green’s Theorem is zero. If Eq. D.3 is evaluated at point \( P \) and \( P \) is in the region \( R_f \), then the resulting expression can be seen in Eq. D.4.

\[
\Phi(P) = -\frac{1}{2\pi} \int_S [\ln r \nabla \Phi - \Phi \nabla \ln r] \cdot \mathbf{n} ds \tag{D.4}
\]

Care must be taken to exclude the point \( P \) from the region of integration, since \( \ln r \) approaches infinity as \( r \) approaches zero. This is accomplished by enclosing \( P \) in a circular boundary of radius \( \epsilon \) and taking the limit as \( \epsilon \) approaches zero. If Eq. D.3 is evaluated at point \( P \) and \( P \) is in the region \( R_i \), then the resulting expression can be seen in Eq. D.5.

\[
0 = -\frac{1}{2\pi} \int_S [\ln r \nabla \Phi_i - \Phi_i \nabla \ln r] \cdot \mathbf{n} ds \tag{D.5}
\]

Add Eq. D.5 to Eq. D.4 noting that for each of these expressions the normal vector points in the opposite direction to get

\[
\Phi(P) = -\frac{1}{2\pi} \int_S [(\nabla \Phi - \nabla \Phi_i) \ln r - (\Phi - \Phi_i) \nabla \ln r] \cdot \mathbf{n} ds \tag{D.6}
\]

The functional form of the integrand in Eq. D.6 is that of a 2-D source and 2-D doublet with strengths \( \sigma \) and \( \mu \) respectively. These strengths are described in Eq. D.7.

\[
-\sigma = \frac{\partial \Phi}{\partial n} - \frac{\partial \Phi_i}{\partial n} \\
-\mu = \Phi - \Phi_i \tag{D.7}
\]
For a modeled exterior flow, the interior boundary, $S$ can be composed of the boundary of a solid surface, $S_B$ and the boundary of a wake, $S_w$. If we ignore entrainment into the wake, then there are no source terms on the wake boundary. At the exterior boundary, $S_\infty$, all of the perturbation potentials approach zero and the potential is equal to $\phi_\infty$. With all of this in mind, Eq. D.6 can be rewritten as Eq. D.8.

$$\Phi(P) = \frac{1}{2\pi} \int_{S_B} [\sigma \ln r - \mu \frac{\partial}{\partial n}(\ln r)] ds - \frac{1}{2\pi} \int_{S_w} \mu_w \frac{\partial}{\partial n}(\ln r)] ds + \phi_\infty(P)$$  \hspace{1cm} (D.8)

The no-penetration boundary condition is imposed implicitly using the Dirichlet type boundary condition. This condition states that setting the total potential in the region $R_i$, $\Phi_i$, equal to a constant is equivalent to imposing no flow through the boundary. The condition can also be imposed by setting $\Phi_i$ equal to $\phi_\infty$ even though the potential of the free stream is not a constant. Therefore, Eq. D.8 reduces to Eq. D.9.

$$\frac{1}{2\pi} \int_{S_B} [\sigma \ln r - \mu \frac{\partial}{\partial n}(\ln r)] ds - \frac{1}{2\pi} \int_{S_w} \mu_w \frac{\partial}{\partial n}(\ln r)] ds = 0$$  \hspace{1cm} (D.9)

The boundary condition is automatically satisfied at the infinity boundary. This is a property of the analytic solutions to Laplace's equation (harmonic functions).

If the interior boundary is discretized into panels, Eq. D.9 can be written as shown in Eq. D.10, where $N_B$ is the number of panels comprising the surface $S_B$ and $N_w$ is the number of panels comprising the wake $S_w$.

$$\sum_{k=1}^{N_B} (\mu_k C_{jk}) + \sum_{k=1}^{N_B} (\sigma_k B_{jk}) + \sum_{k=1}^{N_w} (\mu_w C_{jk}) = 0 |_{j=1,N_B}$$  \hspace{1cm} (D.10)

The coefficients $B$ and $C$ are shown in Eq. D.11.

$$B_{jk} = \frac{1}{2\pi} \int_k \ln r ds$$

$$C_{jk} = -\frac{1}{2\pi} \int_k \frac{\partial}{\partial n}(\ln r) ds$$  \hspace{1cm} (D.11)

The strength of the wake doublets, $\mu_w$, is determined by an auxiliary condition. The source strengths are determined using the Dirichlet boundary condition. Equation D.7 defines the source strength as the dot product of the gradient of the potential jump across a panel with the normal vector of that panel. Since the definition of no-penetration requires $\frac{\partial \Phi}{\partial n} = 0$ and the potential of the interior region is equal to the
free stream potential, the source strength is equal to the normal component of the free stream velocity (Eq. D.12).

\[-\sigma_k = \left( \frac{\partial \Phi}{\partial n} - \frac{\partial \Phi_i}{\partial n} \right)_k = - \frac{\partial \phi_\infty}{\partial n} = -V_\infty \cdot n \]  

(D.12)

This leaves the surface doublet strengths as the $N_B$ unknowns to match the $N_B$ equations.

The coefficients, $B$ and $C$ are the influence coefficients for a distributed source and distributed doublet. These can easily be evaluated in the panel reference frame which is described in Fig. D.2. The influence coefficient $B$ is evaluated in Eq. D.13.

\[
\int_{x_1}^{x_2} \ln r \, dx = \frac{1}{2} \left[ (x - x_1) \ln(r_1^2) - (x - x_2) \ln(r_2^2) - 2(x_2 - x_1) + 2z(\theta_2 - \theta_1) \right] \\
r_1^2 = (x - x_1)^2 + z^2 \\
r_2^2 = (x - x_2)^2 + z^2 \\
\theta_1 = \tan^{-1} \frac{z}{x - x_1} \\
\theta_2 = \tan^{-1} \frac{z}{x - x_2} 
\]  

(D.13)

The influence coefficient $C$ is evaluated in Eq. D.14.

\[
\int_{x_1}^{x_2} \frac{\partial}{\partial n} (\ln r) \, dx = - \frac{\partial}{\partial z} \int_{x_1}^{x_2} (\ln r) \, dx \\
= - [\theta_2 - \theta_1] 
\]  

(D.14)

The negative sign is due to the normal vector pointing in the negative direction. The influence coefficient of the distributed doublet in Eq. D.14 is mathematically identical to that of discrete vortices at the panel nodes. The strength of the discrete vortex at $x_1$ is equal to $\mu$ and that at $x_2$ is equal to $-\mu$.

### D.2 3-D Flows

For the 3-D formulation, set $\Phi_1 = \frac{1}{r}$ and $\Phi_2 = \Phi$. If $P$ is in the region $R_f$, Green’s Theorem yields Eq. D.15. A sphere of radius $\epsilon$ surrounds point $P$ in order to exclude
it from the region of integration. The limit as $\epsilon$ approaches zero is taken just as was done for the 2-D case.

\[ \Phi(P) = \frac{1}{4\pi} \iint_{S} \left[ \frac{1}{r} \nabla \Phi - \Phi \nabla \frac{1}{r} \right] \cdot \mathbf{n} ds \quad \text{(D.15)} \]

If $P$ is in $R_i$, then Green’s Theorem yields

\[ 0 = \frac{1}{4\pi} \iint_{S} \left[ \frac{1}{r} \nabla \Phi_i - \Phi_i \nabla \frac{1}{r} \right] \cdot \mathbf{n} ds \quad \text{(D.16)} \]

Equation D.17 is a result of the summation of Eq. D.15 and D.16.

\[ \Phi(P) = \frac{1}{4\pi} \iint_{S} \left[ \frac{1}{r} \nabla (\Phi - \Phi_i) - (\Phi - \Phi_i) \nabla \frac{1}{r} \right] \cdot \mathbf{n} ds \quad \text{(D.17)} \]

Since the functional forms of the integrand are that of a 3-D source distribution and a 3-D doublet distribution, the strengths of these distributions are the coefficients (Eq. D.7). Equation D.18 shows Eq. D.7 substituted into Eq. D.17.

\[ \Phi(P) = -\frac{1}{4\pi} \iint_{S_B} \left[ \sigma \frac{1}{r} - \mu \nabla \frac{1}{r} \right] \cdot \mathbf{n} ds + \frac{1}{4\pi} \iint_{S_w} \left[ \mu_w \nabla \frac{1}{r} \right] \cdot \mathbf{n} ds + \phi_{\infty}(P) \quad \text{(D.18)} \]

The no-penetration boundary condition is imposed in the same implicit fashion as was done in 2-D in a Dirichlet formulation. The 3-D equivalent to Eq. D.9 is shown in Eq. D.19.

\[ -\frac{1}{4\pi} \iint_{S_B} [\sigma \frac{1}{r} - \mu \frac{\partial}{\partial n} \left( \frac{1}{r} \right)] ds + \frac{1}{4\pi} \iint_{S_w} [\mu_w \frac{\partial}{\partial n} \left( \frac{1}{r} \right)] ds = 0 \quad \text{(D.19)} \]

If the interior boundary is discretized, then Eq. D.19 can be written in discretized form as in Eq. D.10 with the matrix components shown in Eq. D.20 and D.21.

\[ B_{jk} = -\frac{1}{4\pi} \iint_{k} \frac{1}{r} ds \quad \text{(D.20)} \]

\[ C_{jk} = \frac{1}{4\pi} \iint_{k} \frac{\partial}{\partial n} \left( \frac{1}{r} \right) ds \quad \text{(D.21)} \]
Figure D.1: Regions for Green’s Identity Formulation of Panel method
Figure D.2: Description of Panel Reference Frame
Appendix E

Computer Codes

The program, *main*, connects the optimizer to all of the analyses and allows the user to adjust all variables pertaining to the optimization. The user can adjust the design variables, variable bounds, constraints, and DOT software options. Call statements for the analysis codes allow for quick and easy replacement of the individual analyses.

E.1 main.f

**************************************************************************
* Program: mainc.f
* Author : Jason Tyll
* Date : 4/1/97
* purpose: Calls DOT and analysis codes.
* Set up for 7 Design Variable Problem.
**************************************************************************

IMPLICIT DOUBLE PRECISION (A-H,O-Z)
parameter(ndv=7,nrwk=1000,nriwk=300)
dimension iprm(20), iwk(nriwk)
dimension x(ndv), xl(ndv), xu(ndv), rprm(20), wk(nrwk)
dimension xmap(ndv)
dimension g(7)

info=0
method=3
iprint=7
ncon=7
APPENDIX E. COMPUTER CODES

* Design variables and variable limits
  x(1) = 0.042857
  x(2) = -0.05
  x(3) = 0.03683
  x(4) = 1.00105
  x(5) = 0.10619
  x(6) = -8.3216
  x(7) = -4.8250
  x(1) = 0.286
  x(2) = 0.1667
  x(3) = 0.0377
  x(4) = -0.0588
  x(5) = -0.00840
  x(6) = 0.1416
  x(7) = -0.0637
  xu(1) = 1.
  xu(2) = 1.
  xu(3) = 1.
  xu(4) = 1.
  xu(5) = 1.
  xu(6) = 1.
  xu(7) = 1.
  minmax = -1
  do i = 1, 20
    rprm(i) = 0.0
    iprm(i) = 0
  end do
  rprm(7) = 0.05
  iprm(2) = -1

* Transform the design variables to nonscaled values
  (Affine Scaling)
  xmap(1) = 3.5 * x(1)
  xmap(2) = 0.5 + 3.0 * x(2)
  xmap(3) = 14.7 + 35.3 * x(3)
  xmap(4) = 1.111 + 1.887 * x(4)
  xmap(5) = 0.166 + 7.854 * x(5)
  xmap(6) = 21.67 + 2.3298 * x(6)
  xmap(7) = 20.24 + 3.768 * x(7)

* Call the analyses
  call geomc(xmap)
  call xinyout(xmap,ysep1,ysep2)
  call pnlbluffc3(cd,cl,xmap(6),xmap(7),ysep1,ysep2,con1,con2,
                   cdbase,ITRBSEP,npanel,info)
  call mvehicc(vehicmss,wempty)
  call opcst(cd,cl,vehicmss,flevit,encost)
  call lifecost(wempty,encost,clc,coperate,cvehic,cinvest)
  write(*,*) wempty,encost,clc,coperate,cvehic
  write(*,*) i,cd,cl,cm
  write(*,*) i, con1, con2
APPENDIX E. COMPUTER CODES

Call the optimizer (DOT SQP Method)

\[ \text{call DOT(info,method,iprint,ndv,ncon,x,xl,xu,obj,minmax,} \]
\[ \text{g,rprm,iprm,wk,nrwk,iwk,nriwk)} \]

* If the optimization is complete, perform optimization at optimum
* point and exit the program.

\[ \text{if (info.eq.0) then} \]
\[ \text{transform the design variables to unscaled values} \]
\[ \text{(this mapping is repeated for the optimization loop which} \]
\[ \text{begins at line 10)} \]
\[ \text{xmap(1)=3.5*x(1)} \]
\[ \text{xmap(2)=0.5+3.0*x(2)} \]
\[ \text{xmap(3)=14.7+35.3*x(3)} \]
\[ \text{xmap(4)=1.111+1.887*x(4)} \]
\[ \text{xmap(5)=0.166+7.854*x(5)} \]
\[ \text{xmap(6)=21.67+2.3298*x(6)} \]
\[ \text{xmap(7)=20.24+3.768*x(7)} \]
\[ \text{call geomc(xmap)} \]
\[ \text{call xinyout(xmap,ysep1,ysep2)} \]
\[ \text{call pnlbluffc3(cd,cl,xmap(6),xmap(7),ysep1,ysep2,con1,con2,} \]
\[ \text{cdbase,ITRBSEP,npanel,info)} \]
\[ \text{call mvehicc(vehicmss,wempty)} \]
\[ \text{call opcost(cd,cl,vehicmss,flevit,encost)} \]
\[ \text{call lifecost(wempty,encost,clc,coperate,cvehic,cinvest)} \]
\[ \text{obj=abs(cd/cl)} \]
\[ \text{write(*,*) wempty,encost,clc,coperate,cvehic} \]
\[ \text{goto 20} \]
\[ \text{end if} \]

* Calculate the objective function and the constraints.

\[ \text{if (info.eq.1) then} \]
\[ \text{transform the design variables to unscaled values} \]
\[ \text{(this mapping is repeated for the optimization loop which} \]
\[ \text{begins at line 10)} \]
\[ \text{xmap(1)=3.5*x(1)} \]
\[ \text{xmap(2)=0.5+3.0*x(2)} \]
\[ \text{xmap(3)=14.7+35.3*x(3)} \]
\[ \text{xmap(4)=1.111+1.887*x(4)} \]
\[ \text{xmap(5)=0.166+7.854*x(5)} \]
\[ \text{xmap(6)=21.67+2.3298*x(6)} \]
\[ \text{xmap(7)=20.24+3.768*x(7)} \]
\[ \text{call geomc(xmap)} \]
\[ \text{call xinyout(xmap,ysep1,ysep2)} \]
\[ \text{call pnlbluffc3(cd,cl,xmap(6),xmap(7),ysep1,ysep2,con1,con2,} \]
\[ \text{cdbase,ITRBSEP,npanel,info)} \]
\[ \text{call mvehicc(vehicmss,wempty)} \]
\[ \text{call opcost(cd,cl,vehicmss,flevit,encost)} \]
\[ \text{call lifecost(wempty,encost,clc,coperate,cvehic,cinvest)} \]
\[ \text{obj=abs(cd/cl)} \]
\[ g(1) = \frac{xmap(1) - xmap(2)}{xmap(2)} \]
\[ g(2) = -cdbase \]
\[ g(3) = \text{con1} \]
\[ g(4) = -\text{con1} \]
\[ g(5) = \text{con2} \]
\[ g(6) = -\text{con2} \]
\[ g(7) = \text{dble(npanel/4-ITRBSEP)/dble(npanel)} \]
\[ \text{write(*,*) wempty,encost,clc,coperate,cvehic} \]
\[ \text{goto 10} \]
end if

20 continue
stop
end
The subroutine, `geomc` generates the side view geometry for the 2-D Northrop Grumman geometry definition. This definition involves 5 design variables and is identical from front to back.

### E.2 geomc.f

```fortran
subroutine geomc(var)

integer imax, jmax, kmax
double precision l, h
parameter(imax=17,jmax=20,kmax=20,l=.6,h=.35)
double precision xn, xf, f, th, n
double precision x(imax+1), yu(imax+1), yl(imax+1), var(7)
double precision pi,d
integer i

open(unit=9,file='mag.dat',status='unknown')

th=var(3)
n=var(4)
f=var(5)
xn=var(1)
xf=var(2)

pi=4.*atan(1.)
th=th*pi/180.
do i=1,imax+1
     x(i)=l-l*cos(.5*pi*(real(i)-1.)/(real(imax)))
end do

do i=1,imax+1
     if (x(i).le.xn) then
        yu(i)=n*tan(th)/(xn**((1.-n)/n))*x(i)**(1./n)
        yl(i)=-f*n*tan(th)*xn*(1.-(abs(x(i)-xn)/xn)**n)**(1./n)
     end if
     if (x(i).gt.xn .and. x(i).le.xf) then
        yu(i)=n*tan(th)*xn+tan(th)*(x(i)-xn)
        yl(i)=-f*n*tan(th)*xn
```

---

APPENDIX E. COMPUTER CODES

---
APPENDIX E. COMPUTER CODES

```fortran
end if
if (x(i).gt.xf) then
  p=(-2.*(h-f*n*tan(th)*xn)+2.*(n*tan(th)*xn+tan(th)*
  1 (xf-xn))+tan(th)*(l-xf))/(l-xf)**3.
  q=(3.*(h-f*n*tan(th)*xn)-3.*(n*tan(th)*xn+tan(th)*
  1 (xf-xn))-2.*tan(th)*(l-xf))/(l-xf)**2.
  r=tan(th)
  s=n*tan(th)*xn+tan(th)*(xf-xn)
  yu(i)=p*(x(i)-xf)**3.+q*(x(i)-xf)**2.+r*(x(i)-xf)+s
  yl(i)=-f*n*tan(th)*xn
end if
end do

* Output the geometry.
do i=1,imax+1
  write(9,*) 4.*l-x(i), yl(i)+f*n*tan(th)*xn
end do
do i=imax+1,1,-1
  write(9,*) x(i), yl(i)+f*n*tan(th)*xn
end do
do i=1,imax+1
  write(9,*) 4.*l-x(i), yu(i)+f*n*tan(th)*xn
end do
do i=2,imax+1
  write(9,*) x(i), yu(i)+f*n*tan(th)*xn
end do
do i=1,imax
  write(9,*) 1+real(i)/real(imax+1)*2.*1, yu(imax+1)
  +f*n*tan(th)*xn
end do
do i=imax+1,1,-1
  write(9,*) 4.*l-x(i), yu(i)+f*n*tan(th)*xn
end do

close(9)
return
end
```
The aerodynamic analysis using the unsteady vortex panel method for separated flow over bluff bodies in ground effect is performed in subroutine, `pnlbluff`. The user can make adjustments to variables involved in the aerodynamic analysis in the “parameter” statements and immediately following the variable declarations.

**E.3 pnlbluff.f**

```
subroutine pnlbluffc3(cd,cl,xsep1,xsep2,ysep1,ysep2,
                      con1,con2,cdbase,ITRBSEP,n,infodot)

*** Program: pnlbluffc3.f
*** Author: Jason Tyll
*** Date: 4/1/97 (updated to c3 on 4/17/97)
*** Purpose: unsteady vortex panel method
*** (method was developed from Mook, JFE, 1994)
*** For separated flow over bluff bodies.
*** 7 Design variable formulation.
*** Use cubic splines to model piecewise linear pressure
*** distribution for the Stratford Criterion.
***
*** declarations
integer i, j, n, tstep, timemax, tt, t, nmax, info, lwork
integer i1, i2
integer sep1, sep2, nmin, nminest, infodot, ITRBSEP
integer step, tmmax
integer nstag, icon
double precision nu, rho, strat
double precision ratio, time, L, deltt, h
parameter(nmax=200, ratio=1., time=0.43, tmmax=600, L=24., h=3.5)
parameter(nu=1.56E-5, rho=1.177)
parameter(strat=0.39)
double precision pi, alpha, al, gnd, alt
double precision xmom, zmom, sigma(tmmax)
double precision xx(nmax+1), yy(nmax+1)
double precision xxw1(tmmax), yyw1(tmmax)
double precision xxw2(tmmax), yyw2(tmmax)
double precision xxwtmp1(tmmax), yywtmp1(tmmax)
double precision xxwtmp2(tmmax), yywtmp2(tmmax)
double precision delt1(nmax), co(nmax,2)
double precision d1, d2, x, y, delttth, R
double precision v1x(nmax), v1y(nmax), v2x(nmax), v2y(nmax)
double precision vv1x(nmax), vv1y(nmax), vv2x(nmax), vv2y(nmax)
double precision a(nmax+1, nmax), b(nmax,1)
double precision g(nmax), gw1(tmmax), gw2(tmmax), glast(nmax)
```
APPENDIX E. COMPUTER CODES

```fortran

double precision u, v
double precision work(2*nmax+1)
double precision delx, dely, cl, cd, cm, cp(nmax), cf(2000)
double precision cdbase, vel(nmax)
double precision error
double precision uinf
double precision s(nmax+1), dcpds(nmax), c(nmax)
double precision cpmin, reo, xxmin, cfgap, dpdx
double precision ssep1, ssep2, gsep1, gsep2
double precision xsep1, xsep2, ysep1, ysep2
double precision glast_sep1, glast_sep2
double precision atmp(nmax+1, nmax)
double precision clavg, cdavg, cmavg, cpavg(nmax)
double precision clmin, clmax, cdmin, cdmax, cmmin, cmmax
double precision split, combine
double precision phib0(nmax, nmax), phib1(nmax, nmax)
double precision phifs(nmax), phi(nmax), philast(nmax)
double precision phivu(nmax, tmmax), phivl(nmax, tmmax)
double precision con1, con2
double precision fpo, fpn, root1, root2, smin, p0, p1, cpmintry
double precision aa(nmax), bb(nmax), cc(nmax), dd(nmax)
real time1, time2, tarray(2)

open(unit=8, file='cpsd.dat', status='unknown')
open(unit=9, file='mag.dat', status='old')
open(unit=69, file='thist.dat', status='unknown')

pi=4.*atan(1.)

* user input
* The number of panels (must be divisible by 4)
  n=104
* The angle of attack in degrees (must be zero for ground effect)
  alpha=0.
  al=alpha*pi/180.
* Set gnd to 0. for OGE or 1. for IGE
  gnd=1.
* altitude in units of cord
  alt=0.1
* separation panel #1
  sep1=2
* separation panel #2
  sep2=n
* Free Stream Velocity
  uinf=134.

* calculate time increment and number of time steps
  deltt=2.*L/n/uinf/ratio
  timemax=idnint(time/deltt)
```

* user input
* The number of panels (must be divisible by 4)
  n=104
* The angle of attack in degrees (must be zero for ground effect)
  alpha=0.
  al=alpha*pi/180.
* Set gnd to 0. for OGE or 1. for IGE
  gnd=1.
* altitude in units of cord
  alt=0.1
* separation panel #1
  sep1=2
* separation panel #2
  sep2=n
* Free Stream Velocity
  uinf=134.

* calculate time increment and number of time steps
  deltt=2.*L/n/uinf/ratio
  timemax=idnint(time/deltt)
* Initialize Arrays
  do i=1,nmax
    delt1(i)=0.
    co(i,2)=0.
    v1x(i)=0.
    v1y(i)=0.
    v2x(i)=0.
    v2y(i)=0.
    vv1x(i)=0.
    vv1y(i)=0.
    vv2x(i)=0.
    vv2y(i)=0.
    b(i,1)=0.
    g(i)=0.
    glast(i)=0.
    cp(i)=0.
    cpavg(i)=0.
    cf(i)=0.
    vel(i)=0.
    dcpds(i)=0.
    c(i)=0.
  end do
  do i=1,timemax
    xxw1(i)=0.
    xxw2(i)=0.
    yyw1(i)=0.
    yyw2(i)=0.
    gw1(i)=0.
    gw2(i)=0.
    xxwtmp1(i)=0.
    xxwtmp2(i)=0.
    yywtmp1(i)=0.
    yywtmp2(i)=0.
    sigma(i)=0.
  end do
  do i=1,nmax+1
    xx(i)=0.
    yy(i)=0.
    s(i)=0.
  end do
  do i=1,nmax+1
    do j=2,nmax
      a(i,j)=0.
      atmp(i,j)=0.
    end do
  end do
  do i=1,2*nmax+1
    work(i)=0.
  end do
  do i=1,nmax
    do j=1,nmax
      phib0(i,j)=0.
      phib1(i,j)=0.
    end do
  end do
* point about which pitching moment is calculated
  \( x_{mom} = 0 \)
  \( z_{mom} = 0.405 + \text{alt} \)

* zero out the average force & moment coeff
  \( c_{lavg} = 0 \)
  \( c_{davg} = 0 \)
  \( c_{mavg} = 0 \)
  do \( i = 1, n \)
      \( c_{pavg}(i) = 0 \)
  end do

* read in the panel end points
  note: global position variables (\( xx, yy \)), local variables (\( x, y \))
  do \( j = 1, n+1 \)
      read(9,*) \( xx(j), yy(j) \)
  end do
  do \( j = 1, n+1 \)
      \( yy(j) = yy(j) + \text{alt} \)
  end do

* set initial separation points
  \( y_{sep1} = y_{sep1} + \text{alt} \)
  \( y_{sep2} = y_{sep2} + \text{alt} \)
  \( \text{sep1} = 2 \)
  do \( j = 2, n/2 \)
      if (abs(x_{sep1}-xx(j)).lt.abs(x_{sep1}-xx(sep1))) then
          \( \text{sep1} = j \)
      end if
  end do
  \( \text{sep2} = n \)
  do \( j = n/2, n+1 \)
      if (abs(x_{sep2}-xx(j)).lt.abs(x_{sep2}-xx(sep2))) then
          \( \text{sep2} = j \)
      end if
  end do
  \( g_{sep1} = 0 \)
  \( g_{sep2} = 0 \)

* calculate panel lengths
  do \( j = 1, n \)
      \( \text{deltl}(j) = \sqrt{(xx(j+1)-xx(j))^2 + (yy(j+1)-yy(j))^2} \)
end do

* establish surface points (colocation points)
do j=1,n
  co(j,1)=(xx(j)+xx(j+1))/2.
  co(j,2)=(yy(j)+yy(j+1))/2.
end do

****Start Clock****
call etime(tarray)
time1=tarray(1)

* establish influence coefficients
do j=1,n
  phifs(j)=uinf*(xx(j)*cos(al)+yy(j)*sin(al))
do i=1,n

  d1=(xx(i+1)-xx(i))/deltl(i)
  d2=(yy(i+1)-yy(i))/deltl(i)
  x=d1*(co(j,1)-xx(i))+d2*(co(j,2)-yy(i))
  y=-d2*(co(j,1)-xx(i))+d1*(co(j,2)-yy(i))+1.E-12
  deltth=atan(x/y)-atan((x-deltl(i))/y)
  R=((x-deltl(i))**2.+y**2.)/(x**2.+y**2.)
  v1x(i)=1./(2.*pi*deltl(i))*(
    ((deltl(i)-x)*deltth-0.5*y*log(R))
  )
  v1y(i)=1./(2.*pi*deltl(i))*(
    (y*deltth-deltl(i)+0.5*(deltl(i)-x)*log(R))
  )
  v2x(i)=1./(2.*pi*deltl(i))*(
    (x*deltth+0.5*y*log(R))
  )
  v2y(i)=1./(2.*pi*deltl(i))*(
    (deltl(i)-y*deltth+0.5*x*log(R))
  )
  vv1x(i)=d1*v1x(i)-d2*v1y(i)
  vv1y(i)=d2*v1x(i)+d1*v1y(i)
  vv2x(i)=d1*v2x(i)-d2*v2y(i)
  vv2y(i)=d2*v2x(i)+d1*v2y(i)
  phib0(j,i)=-1./(2.*pi)*(
    (x*atan(y/x)-(x-deltl(i)))*
    atan(y/(x-deltl(i)))-0.5*y*log(R))
  )
  phib1(j,i)=-1./(2.*pi)*(
    (-x*y*0.5*log(R)-0.5*y*deltl(i)
  )+((x**2.-y**2.)*0.5*atan(y/x)-
  1
    (x**2.-deltl(i)**2.-y**2.)*0.5*atan(y/
  1
    (x-deltl(i))))
end do

do i=2,n
  a(j,i)=-(vv2x(i-1)+vv1x(i))*(yy(j+1)-yy(j))/deltl(j)
  +((vv2y(i-1)+vv1y(i))*(xx(j+1)-xx(j))/deltl(j)
1
end do

a(j,1)=-(vv2x(n)+vv1x(1))*(yy(j+1)-yy(j))/deltl(j)
+ (vv2y(n)+vv1y(1))*(xx(j+1)-xx(j))/deltl(j)

if (gnd.eq.1.) then
  do i=1,n
    d1=(xx(i+1)-xx(i))/deltl(i)
d2=(yy(i+1)-yy(i))/deltl(i)

    x=d1*(co(j,1)-xx(i))+d2*(co(j,2)+yy(i))
y=-d2*(co(j,1)-xx(i))+d1*(co(j,2)+yy(i))
deltth=atan(x/y)-atan((x-deltl(i))/y)
R=((x-deltl(i))**2.+y**2.)/(x**2.+y**2.)

    v1x(i)=1./(2.*pi*deltl(i))*
      ((deltl(i)-x)*deltth-0.5*y*log(R))
    v1y(i)=1./(2.*pi*deltl(i))*
      (y*deltth-deltl(i)+0.5*(deltl(i)-x)*log(R))
    v2x(i)=1./(2.*pi*deltl(i))*
      (deltl(i)-y*deltth+0.5*x*log(R))
    v2y(i)=1./(2.*pi*deltl(i))*
      (deltl(i)-y*deltth+0.5*x*log(R))

    vv1x(i)=d1*v1x(i)-d2*v1y(i)
    vv1y(i)=d2*v1x(i)+d1*v1y(i)
    vv2x(i)=d1*v2x(i)-d2*v2y(i)
    vv2y(i)=d2*v2x(i)+d1*v2y(i)
  end do

  do i=2,n
    a(j,i)=a(j,i)+(vv2x(i-1)+vv1x(i))*(yy(j+1)-yy(j))/deltl(j)
- (vv2y(i-1)+vv1y(i))*(xx(j+1)-xx(j))/deltl(j)
  end do

a(j,1)=a(j,1)+(vv2x(n)+vv1x(1))*(yy(j+1)-yy(j))/deltl(j)
- (vv2y(n)+vv1y(1))*(xx(j+1)-xx(j))/deltl(j)

  end if
end do

  * establish constraint equation
  do i=2,n
    a(n+1,i)=0.5*(deltl(i-1)+deltl(i))
  end do
a(n+1,1)=0.5*(deltl(n)+deltl(1))
* begin time steps
  do tstep=2, timemax
* calculate sigma (vortex blob diameter)
sigma(tstep)=1.75*uinf*deltt
* update sigma values
  if (tstep.gt.2) then
    do t=2, tstep-1
      sigma(t)=sqrt(sigma(t)**2.+4.*nu*deltt)
    end do
  end if
* establish strength of shed wake
  gw1(tstep)=-(gsep1**2.)*deltt/2.
gw2(tstep)=(gsep2**2.)*deltt/2.
* location of the shed wake
  yyw1(tstep)=ysep1
  yyw2(tstep)=ysep2
  if (tstep.eq.3) then
    xxw1(tstep)=xsep1+0.1*deltl(sep1)
    xxw2(tstep)=xsep2+0.1*deltl(sep2)
  else
    xxw1(tstep)=xsep1+dabs(gsep1)*deltt/5.
    xxw2(tstep)=xsep2+dabs(gsep2)*deltt/5.
  end if
* Establish RHS
  do j=1, n
    b(j,1)=uinf*(dcos(al)*(yy(j+1)-yy(j))-dsin(al)*
    *(xx(j+1)-xx(j)))/deltl(j)
    if (tstep.le.200) then
      do t=2, tstep
        b(j,1)=b(j,1)+1./(2.*pi)/deltl(j)/((co(j,1)-xxw1(t))**2.+((co(j,2)-yyw1(t))**2.+sigma(t)**2.)*((co(j,2)-
        yyw1(t))*(yy(j+1)-yy(j))+co(j,1)-xxw1(t)))*
        (xx(j+1)-xx(j)))*gw1(t)
        b(j,1)=b(j,1)+1./(2.*pi)/deltl(j)/((co(j,1)-xxw2(t))**2.+sigma(t)**2.)*((co(j,2)-
        yyw2(t))*(yy(j+1)-yy(j))+co(j,1)-xxw2(t)))*
        (xx(j+1)-xx(j)))*gw2(t)
        b(j,1)=b(j,1)-gnd/(2.*pi)/deltl(j)/((co(j,1)-xxw1(t))**2.+((co(j,2)+yyw1(t))**2.+sigma(t)**2.)*((co(j,2)+
        yyw1(t))*(yy(j+1)-yy(j))+co(j,1)-xxw1(t)))*
        (xx(j+1)-xx(j)))*gw1(t)
      end do
    end if
  end do
end do 
else
  do t=tstep-199,tstep
    b(j,1)=b(j,1)+1./(2.*pi)/deltl(j)/((co(j,1)-xxw1(t))**2.+ (co(j,2)-yyw1(t))**2.+sigma(t)**2.)*((co(j,2)-1 yyw1(t))*(yy(j+1)-yy(j))+(co(j,1)-xxw1(t))*1 (xx(j+1)-xx(j)))*gw1(t)
    b(j,1)=b(j,1)+1./(2.*pi)/deltl(j)/((co(j,1)-xxw2(t))**2.+ (co(j,2)-yyw2(t))**2.+sigma(t)**2.)*((co(j,2)-1 yyw2(t))*(yy(j+1)-yy(j))+(co(j,1)-xxw2(t))*1 (xx(j+1)-xx(j)))*gw2(t)
    b(j,1)=b(j,1)-gnd/(2.*pi)/deltl(j)/((co(j,1)-xxw1(t))**2.+ (co(j,2)+yyw1(t))**2.+sigma(t)**2.)*((co(j,2)+1 yyw1(t))*(yy(j+1)-yy(j))+(co(j,1)-xxw1(t))*1 (xx(j+1)-xx(j)))*gw1(t)
    b(j,1)=b(j,1)-gnd/(2.*pi)/deltl(j)/((co(j,1)-xxw2(t))**2.+ (co(j,2)+yyw2(t))**2.+sigma(t)**2.)*((co(j,2)+1 yyw2(t))*(yy(j+1)-yy(j))+(co(j,1)-xxw2(t))*1 (xx(j+1)-xx(j)))*gw2(t)
  end do
end if
end do

b(n+1,1)=0.
do t=2,tstep
  b(n+1,1)=b(n+1,1)-gw1(t)-gw2(t)
end do

* glast will be used to calc the dynamic term in cp equation
  do i=1,n
    glast(i)=g(i)
    philast(i)=phi(i)
  end do

  do j=1,n+1
    do i=1,n
      atmp(j,i)=a(j,i)
    end do
  end do

* solve system of equations using a linear least squares
  optimization
  lwork=2*nmax+1
  call DGELS('N',n+1,n,1,atmp,nmax+1,b,nmax+1,work,lwork,info)
  do i=1,n
    g(i)=b(i,1)
    phi(i)=phifs(i)
  end do
  do j=1,n
    do i=1,n-1
      phi(j)=phi(j)+g(i)*phib0(j,i)+g(i+1)*phib1(j,i)
end do
phi(j)=phi(j)+g(n)*phib0(j,n)+g(1)*phib1(j,n)
end do

* calculate gsep at prescribed xsep location
do i=1,n/2-1
  if (xsep1 .ge. xx(i+1) .and. xsep1 .le. xx(i)) then
    gsep1=g(i+1)+(xsep1-xx(i+1))/(xx(i)-xx(i+1))*(g(i)-g(i+1))
  end if
end do
do i=n/2,n
  if (xsep2 .le. xx(i+1) .and. xsep2 .ge. xx(i)) then
    gsep2=g(i)+(xsep2-xx(i))/(xx(i+1)-xx(i))*(g(i+1)-g(i))
  end if
end do

****** update shed vorticy locations for bottom vortex sheet *****
if (tstep .le. 200) then
  i1=2
  i2=tstep
else
  i1=tstep-199
  i2=tstep
end if
do t=i1,i2
  u=0.
v=0.
  * velocity induced by free stream
  u=uinf*dcos(al)
v=uinf*dsin(al)
  * velocity induced by airfoil
  do i=1,n
    d1=(xx(i+1)-xx(i))/deltl(i)
    d2=(yy(i+1)-yy(i))/deltl(i)
    x=d1*(xxw1(t)-xx(i))+d2*(yyw1(t)-yy(i))
    y=-d2*(xxw1(t)-xx(i))+d1*(yyw1(t)-yy(i))+1.E-12
    deltth=atan(x/y)-atan((x-deltl(i))/y)
    R=((x-deltl(i))**2.+y**2.)/(x**2.+y**2.)
    v1x(i)=1./(2.*pi*deltl(i))**
      1 ((deltl(i)-x)*deltth-0.5*y*log(R))
    v1y(i)=1./(2.*pi*deltl(i))**
      1 (y*deltth-deltl(i)+0.5*(deltl(i)-x)*log(R))
    v2x(i)=1./(2.*pi*deltl(i))**
      1 ((x*deltth+deltl(i))*(x*deltth+0.5*y*log(R))
    v2y(i)=1./(2.*pi*deltl(i))**
      1 (deltl(i)-y*deltth+0.5*x*log(R))
    vv1x(i)=d1*v1x(i)-d2*v1y(i)
    vv1y(i)=d2*v1x(i)+d1*v1y(i)
    vv2x(i)=d1*vv2x(i)+d1*vv2y(i)
    vv2y(i)=d1*vv2x(i)-d2*vv2y(i)
APPENDIX E. COMPUTER CODES

vv2y(i)=d2*v2x(i)+d1*v2y(i)

if (i.eq.n) then
  u=u+g(i)*vv1x(i)+g(1)*vv2x(i)
  v=v+g(i)*vv1y(i)+g(1)*vv2y(i)
end if
if (i.ne.n) then
  u=u+g(i)*vv1x(i)+g(i+1)*vv2x(i)
  v=v+g(i)*vv1y(i)+g(i+1)*vv2y(i)
end if

phivl(i,t)=-1./(2.*pi)*atan((yy(i)-yyw1(t))/(xx(i)-xxw1(t)))
end do

if (gnd.eq.1.) then
  do i=1,n
    d1=(xx(i+1)-xx(i))/deltl(i)
    d2=(yy(i+1)-yy(i))/deltl(i)
    x=d1*(xxw1(t)-xx(i))+d2*(yyw1(t)+yy(i))
    y=-d2*(xxw1(t)-xx(i))+d1*(yyw1(t)+yy(i))
    delth=atan(x/y)-atan((x-deltl(i))/y)
    R=((x-deltl(i))**2.+y**2.)/(x**2.+y**2.)
    v1x(i)=1./(2.*pi*deltl(i))*
    (1*(deltl(i)-x)*delth-0.5*y*log(R))
    v1y(i)=1./(2.*pi*deltl(i))*
    (1*(y*delth-deltl(i)+0.5*(deltl(i)-x)*log(R))
    v2x(i)=1./(2.*pi*deltl(i))*(x*delth+0.5*y*log(R))
    v2y(i)=1./(2.*pi*deltl(i))*
    (1*(deltl(i)-y*delth+0.5*x*log(R))
  vv1x(i)=d1*vv1x(i)-d2*vv1y(i)
  vv1y(i)=d2*vv1x(i)+d1*vv1y(i)
  vv2x(i)=d1*vv2x(i)-d2*vv2y(i)
  vv2y(i)=d2*vv2x(i)+d1*vv2y(i)
  if (i.eq.n) then
    u=u-g(i)*vv1x(i)-g(1)*vv2x(i)
    v=v-g(i)*vv1y(i)-g(1)*vv2y(i)
  end if
  if (i.ne.n) then
    u=u-g(i)*vv1x(i)-g(i+1)*vv2x(i)
    v=v-g(i)*vv1y(i)-g(i+1)*vv2y(i)
  end if
end do
end if

velocity induced by wake
if (tstep.le.200) then
  do tt=2, tstep
    if(tt.ne.t) then
      u=u+gw1(tt)/(2.*pi)*(yyw1(t)-yyw1(tt))/
  end do
end if
\begin{verbatim}
APPENDIX E. COMPUTER CODES

1  ((xxw1(t)-xxw1(tt))**2.+((yyw1(t)-yyw1(tt))**2. +
1  +sigma(tt)**2.)+gw2(tt)/(2.*pi)*(yyw1(t)-yyw2(tt))/
1  ((xxw1(t)-xxw2(tt))**2.+((yyw1(t)-yyw2(tt))**2. +
1  +sigma(tt)**2.)
1  u=u-gnd*gw1(tt)/(2.*pi)*(yyw1(t)+yyw1(tt))/
1  ((xxw1(t)-xxw1(tt))**2.+((yyw1(t)+yyw1(tt))**2. +
1  +sigma(tt)**2.)
1  +sigma(tt)**2.)

1  v=-gw1(tt)/(2.*pi)*(xxw1(t)-xxw1(tt))/
1  ((xxw1(t)-xxw1(tt))**2.+((yyw1(t)-yyw1(tt))**2. +
1  +sigma(tt)**2.)
1  +sigma(tt)**2.)
1  v=v+gnd*gw1(tt)/(2.*pi)*(xxw1(t)-xxw1(tt))/
1  ((xxw1(t)-xxw1(tt))**2.+((yyw1(t)+yyw1(tt))**2. +
1  +sigma(tt)**2.)
1  +sigma(tt)**2.)

else
  u=u
  v=v
end if
end do
else
tt=tstep-199,tstep
if(tt.ne.t) then
  u=u+gw1(tt)/(2.*pi)*(yyw1(t)-yyw1(tt))/
  ((xxw1(t)-xxw1(tt))**2.+((yyw1(t)-yyw1(tt))**2. +
  +sigma(tt)**2.)
  +sigma(tt)**2.)
  u=u-gnd*gw1(tt)/(2.*pi)*(yyw1(t)+yyw1(tt))/
  ((xxw1(t)-xxw1(tt))**2.+((yyw1(t)+yyw1(tt))**2. +
  +sigma(tt)**2.)
  +sigma(tt)**2.)
  v=v-gw1(tt)/(2.*pi)*(xxw1(t)-xxw1(tt))/
  ((xxw1(t)-xxw1(tt))**2.+((yyw1(t)-yyw1(tt))**2. +
  +sigma(tt)**2.)
  +sigma(tt)**2.)
  v=v-gw2(tt)/(2.*pi)*(xxw1(t)-xxw2(tt))/
  ((xxw1(t)-xxw2(tt))**2.+((yyw1(t)+yyw2(tt))**2. +
  +sigma(tt)**2.)
  +sigma(tt)**2.)
else
  u=u
  v=v
end if
\end{verbatim}
APPENDIX E. COMPUTER CODES

end do
end if

xxwtmp1(t)=xxw1(t)+u*deltt
yywtmp1(t)=yyw1(t)+v*deltt

end do

**********update shed vorticy locations for top vortex sheet*****

if (tstep.le.200) then
    i1=2
    i2=tstep
else
    i1=tstep-199
    i2=tstep
end if
do t=i1,i2

u=0.
v=0.

* velocity induced by free stream
u=uinf*dcos(al)
v=uinf*dsin(al)

* velocity induced by airfoil
do i=1,n
    d1=(xx(i+1)-xx(i))/deltl(i)
    d2=(yy(i+1)-yy(i))/deltl(i)
    x=d1*(xxw2(t)-xx(i))+d2*(yyw2(t)-yy(i))
    y=-d2*(xxw2(t)-xx(i))+d1*(yyw2(t)-yy(i))+1.E-12
    delth=atan(x/y)-atan((x-deltl(i))/y)
    R=((x-deltl(i))**2.+y**2.)/(x**2.+y**2.)
    v1x(i)=1./(2.*pi*deltl(i))*
        1 (((deltl(i)-x)*deltlth-0.5*y*log(R))
        v1y(i)=1./(2.*pi*deltl(i))*
        1 (y*deltlth-deltl(i)+0.5*(deltl(i)-x)*log(R))
    v2x(i)=1./(2.*pi*deltl(i))*
        1 (x*deltlth+0.5*y*log(R))
    v2y(i)=1./(2.*pi*deltl(i))*
        1 (deltl(i)-y*deltlth+0.5*x*log(R))
    vv1x(i)=d1*v1x(i)-d2*v1y(i)
    vv1y(i)=d2*v1x(i)+d1*v1y(i)
    vv2x(i)=d1*v2x(i)-d2*v2y(i)
    vv2y(i)=d2*v2x(i)+d1*v2y(i)
endif
if (i.eq.n) then
    u=u+g(i)*vv1x(i)+g(1)*vv2x(i)
    v=v+g(i)*vv1y(i)+g(1)*vv2y(i)
endif
if (i.ne.n) then
    u=u+g(i)*vv1x(i)+g(i+1)*vv2x(i)
endif
\[ v = v + g(i) * v v 1 y(i) + g(i+1) * v v 2 y(i) \]
end if

\[ \text{phi}_v(i,t) = -\frac{1}{2 \pi} \arctan\left( \frac{y y(i) - y yw_2(t)}{x x(i) - x xw_2(t)} \right) \]
end do

if (gnd.eq.1.) then
  do i=1,n
    d1=(xx(i+1)-xx(i))/deltl(i)
d2=(yy(i+1)-yy(i))/deltl(i)
x=d1*(xxw2(t)-xx(i))+d2*(yyw2(t)+yy(i))
y=-d2*(xxw2(t)-xx(i))+d1*(yyw2(t)+yy(i))
deltth=atan(x/y)-atan((x-deltl(i))/y)
R=((x-deltl(i))**2.+y**2.)/(x**2.+y**2.)
  end do
if (i.eq.n) then
  u=u-g(i)*vv1x(i)-g(1)*vv2x(i)
v=v-g(i)*vv1y(i)-g(1)*vv2y(i)
end if
if (i.ne.n) then
  u=u-g(i)*vv1x(i)-g(i+1)*vv2x(i)
v=v-g(i)*vv1y(i)-g(i+1)*vv2y(i)
end if
end do

* velocity induced by wake
if (tstep.le.200) then
  do tt=2, tstep
    if (tt.ne.t) then
      u=u+gw1(tt)/(2.*pi)*(yyw2(t)-yyw1(tt))/
      1 ((xxw2(t)-xxw1(tt))**2.+(yyw2(t)-yyw1(tt))**2.1
      1 +sigma(tt)**2.)*gw2(tt)/(2.*pi)*(yyw2(t)-yyw2(tt))/
      1 ((xxw2(t)-xxw2(tt))**2.+(yyw2(t)-yyw2(tt))**2.1
      1 +sigma(tt)**2.)
      u=u-gnd*gw1(tt)/(2.*pi)*(yyw2(t)+yyw1(tt))/
      1 ((xxw2(t)-xxw1(tt))**2.+(yyw2(t)+yyw1(tt))**2.1
      1 +sigma(tt)**2.)*gnd*gw2(tt)/(2.*pi)*(yyw2(t)+yyw2(tt))/
    end if
  end do
  if (i.eq.n) then
    u=u-g(i)*vv1x(i)-g(1)*vv2x(i)
v=v-g(i)*vv1y(i)-g(1)*vv2y(i)
end if
if (i.ne.n) then
  u=u-g(i)*vv1x(i)-g(i+1)*vv2x(i)
v=v-g(i)*vv1y(i)-g(i+1)*vv2y(i)
end if
end do
APPENDIX E. COMPUTER CODES

1  \((xxw2(t)-xxw2(tt))**2.+(yyw2(t)+yyw2(tt))**2. +\sigma(tt)**2.\)
2  \(v=v-gw1(tt)/(2.*\pi)*(xxw2(t)-xxw1(tt))/\)
3  \((xxw2(t)-xxw1(tt))**2.+(yyw2(t)-yyw1(tt))**2. +\sigma(tt)**2.\)
4  \(v=v+gnd*gw1(tt)/(2.*\pi)*(xxw2(t)-xxw1(tt))/\)
5  \((xxw2(t)-xxw1(tt))**2.+(yyw2(t)+yyw1(tt))**2. +\sigma(tt)**2.\)

else
6  \(u=u\)
7  \(v=v\)
end if
end do
else
tt=tstep-199,tstep
if(tt.ne.t) then
  \(u=u+gw1(tt)/(2.*\pi)*(yyw2(t)-yyw1(tt))/\)
1  \((xxw2(t)-xxw1(tt))**2.+(yyw2(t)-yyw1(tt))**2. +\sigma(tt)**2.\)
2  \(u=u-gnd*gw1(tt)/(2.*\pi)*(yyw2(t)+yyw1(tt))/\)
3  \((xxw2(t)-xxw1(tt))**2.+(yyw2(t)+yyw1(tt))**2. +\sigma(tt)**2.\)
4  \(v=v-gw1(tt)/(2.*\pi)*(xxw2(t)-xxw1(tt))/\)
5  \((xxw2(t)-xxw1(tt))**2.+(yyw2(t)-yyw1(tt))**2. +\sigma(tt)**2.\)
6  \(v=v+gnd*gw1(tt)/(2.*\pi)*(xxw2(t)-xxw1(tt))/\)
7  \((xxw2(t)-xxw1(tt))**2.+(yyw2(t)+yyw1(tt))**2. +\sigma(tt)**2.\)

else
  \(u=u\)
  \(v=v\)
end if
end do

xxwtmp2(t)=xxw2(t)+u*deltt
yywtmp2(t)=yyw2(t)+v*deltt
end do
do t=1,tstep
    xxw1(t)=xxwtmp1(t)
    yyw1(t)=yywtmp1(t)
    xxw2(t)=xxwtmp2(t)
    yyw2(t)=yywtmp2(t)
end do

do j=1,n
do t=1,tstep
    phi(j)=phi(j)+gw1(t)*phivl(j,t)+gw2(t)*phivu(j,t)
end do
end do

***** update separation location using the Stratford Criteria
s(1)=0.0
do i=2,n+1
    s(i)=s(i-1)+SQRT((xx(i)-xx(i-1))**2.+(yy(i)-yy(i-1))**2.)
end do
do i=1,n
    cp(i)=1.-g(i)**2./uinf**2.
end do
cp(n+1)=cp(1)
fpo=(cp(2)-cp(1))/(s(2)-s(1))
fpn=(cp(n+1)-cp(n))/(s(n+1)-s(n))
call spline(s,cp,fp,fn,aa,bb,cc,dd,n)
cpmin=1.
* calc estimate for nmin
do i=3*n/4,n
    if (cp(i).lt.cpmin) then
        cpmin=cp(i)
        nminest=i
        nmin=nminest
    end if
end do
end do

do i=nminest-2,n
if (((2.*cc(i))**2.-4.*3.*dd(i)*bb(i)).ge. 0.) then
    root1=(-2.*cc(i)+((2.*cc(i))**2.-4.*3.*dd(i)*bb(i))**0.5)1/(2.*3.*dd(i))
    root2=(-2.*cc(i)-((2.*cc(i))**2.-4.*3.*dd(i)*bb(i))**0.5)1/(2.*3.*dd(i))
    root1=root1+s(i)
    root2=root2+s(i)
    if ((root1.ge.s(i) .and. root1.le.s(i+1)) then
        cpmintry=aa(i)+bb(i)*(root1-s(i))+cc(i)*(root1-s(i))**2.
        +dd(i)*(root1-s(i))**3.
        if(cpmintry.lt.cpmin) then
            cpmin=cpmintry
            smin=root1
            nmin=i+1
        goto 35
end if
end if
end do
APPENDIX E. COMPUTER CODES

end if
end if
if (root2.ge.s(i) .and. root2.le.s(i+1)) then
   cpmintry=aa(i)+bb(i)*(root2-s(i))+cc(i)*(root2-s(i))**2.
      +dd(i)*(root2-s(i))**3.
   if (cpmintry.lt.cpmin) then
      cpmin=cpmintry
      smin=root2
      nmin=i+1
      goto 35
   end if
end if
end if
end if
end do

35 continue

glast_sep1=gsep1
glast_sep2=gsep2

p0=s(nmin+1)
p1=s(n+1)
call bisection(p0,p1,s,aa,bb,cc,dd,
   1 ssep2,uinf,nu,strat,smin,cpmin,n,icon,1.)

* calculate constraint
   con2=xsep2-(xx(icon)+(ssep2-s(icon))/(s(icon+1)-s(icon))
      *(xx(icon+1)-xx(icon)))

   cpmin=1.
   * calc estimate for nmin
   do i=1,n/4
      if (cp(i).lt.cpmin) then
         cpmin=cp(i)
         nminest=i
         nmin=nminest
      end if
   end do

   cpmin=0.
   do i=1,nminest+2
      if (((2.*cc(i))**2.-4.*3.*dd(i)*bb(i)).ge. 0.) then
         root1=(-2.*cc(i)+((2.*cc(i))**2.-4.*3.*dd(i)*bb(i))**0.5)
            /(2.*3.*dd(i))
         root2=(-2.*cc(i)-((2.*cc(i))**2.-4.*3.*dd(i)*bb(i))**0.5)
         root1=root1+s(i)
         root2=root2+s(i)
         if (root1.ge.s(i) .and. root1.le.s(i+1)) then
            cpmintry=aa(i)+bb(i)*(root1-s(i))+cc(i)*(root1-s(i))**2.
               +dd(i)*(root1-s(i))**3.
            if (cpmintry.lt.cpmin) then
               cpmin=cpmintry
               smin=root1
               nmin=i
            end if
         end if
      end if
   end do
**APPENDIX E. COMPUTER CODES**

```fortran
end if
end if
if (root2.ge.s(i) .and. root2.le.s(i+1)) then
  cpmintry=aa(i)+bb(i)*(root2-s(i))+cc(i)*(root2-s(i))**2.
    +dd(i)*(root2-s(i))**3.
  if(cpmintry.lt.cpmin) then
    cpmin=cpmintry
    smin=root2
    nmin=1
  goto 45
  end if
end if
end do

45 continue

p0=s(nmin-1)
p1=s(1)
call bisection(p0,p1,s,aa,bb,cc,dd,
               ssep1,uinf,nu,strat,smin,cpmin,n,icon,-1.)

* calculate constraint
  con1=xsep1-(xx(icon)+(s(icon)-ssep1)/(s(icon)-s(icon-1))
  *(xx(icon-1)-xx(icon)))

* write(*,*) 'xsep1=',xsep1,'xsep2=',xsep2
* write(*,*) 'ysep1=',ysep1,'ysep2=',ysep2
* write(*,*) 'sep1=',sep1,'sep2=',sep2
* write(*,*) 'con1=',con1,'con2=',con2
* write(*,*) 'Constraints in pnlbluffc',con1,con2

************ calculate cp, cl, cd, cm at each time step
cl=0.0
cd=0.0
cm=0.0
if (tstep.gt. timemax-75) then
  do i=1,n
    if (i.eq.1) then
      delx=co(i,1)-co(n,1)
      dely=co(i,2)-co(n,2)
    else
      delx=co(i,1)-co(i-1,1)
      dely=co(i,2)-co(i-1,2)
    end if

    * pressure coeff equation (w/ base pressure model)
    if (i.ge.sep1 .and. i.1e.sep2) then
      cp(i)=1.-g(i)**2./uinf**2.
        -2./uinf**2.*(phi(i)-philast(i))/deltt
      cpavg(i)=cpavg(i)+cp(i)
    else
      cp(i)=1.-gsep2**2./uinf**2.
        -2./uinf**2.*0.5*(phi(sep2)-philast(sep2)+
```


\begin{verbatim}
1   phi(sep1)-philast(sep1))/deltt
cpavg(i)=cpavg(i)+cp(i)
end if

if (i.eq.1) then
  cl=cl-(cp(i)+cp(n))/2.*((xx(i)-xx(n))*cos(al)+
  1   (yy(i)-yy(n))*sin(al))/h
  cd=cd+(cp(i)+cp(n))/2.*((yy(i)-yy(n))*cos(al)-
  1   (xx(i)-xx(n))*sin(al))/h
  cm=cm+(cp(i)+cp(n))/2.*((xx(i)-xx(n))*(co(n,1)-xmom)+
  1   (yy(i)-yy(n))*(co(n,2)-zmom))/h
else
  cl=cl-(cp(i)+cp(i-1))/2.*((xx(i)-xx(i-1))*cos(al)+
  1   (yy(i)-yy(i-1))*sin(al))/h
  cd=cd+(cp(i)+cp(i-1))/2.*((yy(i)-yy(i-1))*cos(al)-
  1   (xx(i)-xx(i-1))*sin(al))/h
  cm=cm+(cp(i)+cp(i-1))/2.*((xx(i)-xx(i-1))*
  1   (co(i-1,1)-xmom)+(yy(i)-yy(i-1))*(co(i-1,2)-zmom))/h
end if

end do
clavg=clavg+cl
cdavg=cdavg+cd
cmavg=cmavg+cm

end if
write(69,*) tstep,cl,cd

***** end time step ****
end do

cl=clavg/75.
cd=cdavg/75.
cm=cmavg/75.
cdbase=cd
write(*,*) 'Cl=',cl,'Cd=',cd,'Cm=',cm

do i=1,n
  cp(i)=cpavg(i)/75.
end do

* output to file
if (infodot.eq.0) then
  write(8,*) 'variables="x","y"
  write(8,*) 'zone f=point
  do i=1,n
    write(8,*) xx(i),',cp(i)
  end do
\end{verbatim}
end if

*    goto 182

***** perform viscous correction calculations****
do  i=1,n+1
    if (cp(i).gt. 1.0) then
        cp(i)=1.0
    end if
end do
call visctop(cf,ITRBSEP,nstag,xx,yy,cp,n,L,uinf,nu)
write(*,*) 'ITRBSEP=',ITRBSEP,'nstag=',nstag
do  j=1,ITRBSEP
    i=nstag-1+j
    if (i.eq.1) then
        delx=co(i,1)-co(n,1)
dely=co(i,2)-co(n,2)
    else
        delx=co(i,1)-co(i-1,1)
        dely=co(i,2)-co(i-1,2)
    end if
    cl=cl+cf(j)*(dely*cos(al)-delx*sin(al))/h
    cd=cd+cf(j)*(dely*sin(al)+delx*cos(al))/h
    cm=cm+cf(j)*(delx*(co(i,2)-zmom)-dely*(co(i,1)-xmom))/h
end do
if (gnd .eq. 0.) then
call viscbttm(cf,ITRBSEP,nstag,xx,yy,cp,n,L,uinf,nu)
write(*,*) 'ITRBSEP=',ITRBSEP,'nstag=',nstag
    do  j=1,ITRBSEP
        i=nstag+1-j
        if (i.eq.1) then
            delx=co(i,1)-co(n,1)
            dely=co(i,2)-co(n,2)
        else
            delx=co(i,1)-co(i-1,1)
            dely=co(i,2)-co(i-1,2)
        end if
        cl=cl+cf(j)*(-dely*cos(al)+delx*sin(al))/h
        cd=cd+cf(j)*(-dely*sin(al)-delx*cos(al))/h
        cm=cm+cf(j)*(delx*(co(i,2)-zmom)+dely*(co(i,1)-xmom))/h
    end do
else
    cpmin=cp(nstag)
    do  j=nstag,1,-1
        if (cp(j).lt.cpmin) then
            cpmin=cp(j)
            xxmin=xx(j)
        end if
    end do
    dpdx=(cpmin-cp(nstag))/(xxmin-xx(nstag))*0.5*rho*uinf**2.
call couette(dpdx,-uinf,alt,cfgap)
do  j=nstag,1,-1
APPENDIX E. COMPUTER CODES

```fortran
if (j.eq.1) then
  delx = co(j,1) - co(n,1)
  dely = co(j,2) - co(n,2)
else
  delx = co(j,1) - co(j-1,1)
  dely = co(j,2) - co(j-1,2)
end if
cl = cl + cfgap * (-dely*cos(al) + delx*sin(al))/h
cd = cd + cfgap * (-dely*sin(al) - delx*cos(al))/h
cm = cm + cfgap * (delx*(co(j,2)-zmom) + dely*(co(j,1)-xmom))/h
end do
end if
continue

if (infodot.eq.0) then
  write(8,'(a)') 'zone'
do i = 1, n+1
    write(8,*) xx(i), yy(i)
  end do
write(8,'(a)') 'zone'
do t = 2, timemax
    write(8,*) xxw1(t), yyw1(t)
  end do
write(8,'(a)') 'zone'
do t = 2, timemax
    write(8,*) xxw2(t), yyw2(t)
  end do
write(*,'(a)') 'Cl=', cl, 'Cd=', cd, 'Cm=', cm
call etime(tarray)
time2 = tarray(1)
write(*,'(a,f10.5)') time1, time2, time2-time1
close(9)
close(69)
return
end

subroutine spline(s,cp,fpo,fpn,aa,bb,cc,dd,n)
**************************************************************** Date: 4/17/97
* Reference: Numerical Analysis by Burden & Faires
* Purpose: Calculates Clamped Cubic Spline
****************************************************************
integer i,j,n,nmax
parameter(nmax=200)
double precision s(nmax+1),aa(nmax),bb(nmax),cc(nmax),dd(nmax)
```
double precision h(nmax), alp(nmax), l(nmax), mu(nmax)
double precision z(nmax), cp(nmax)
double precision fpo, fpn

do i=1, n+1
   aa(i) = cp(i)
end do

do i=1, n
   h(i) = s(i+1) - s(i)
end do

alp(1) = 3. * (aa(2) - aa(1)) / h(1) - 3. * fpo
alp(n+1) = 3. * fpn - 3. * (aa(n+1) - aa(n)) / h(n)

do i=2, n
   alp(i) = 3. / h(i) * (aa(i+1) - aa(i)) - 3. / (i-1) * (aa(i) - aa(i-1))
end do

l(1) = 2. * h(1)
mu(1) = 0.5
z(1) = alp(1) / l(1)

do i=2, n
   l(i) = 2. * (s(i+1) - s(i-1)) - h(i-1) * mu(i-1)
   mu(i) = h(i) / l(i)
   z(i) = (alp(i) - h(i-1) * z(i-1)) / l(i)
end do

l(n+1) = h(n) * (2. * mu(n))
z(n+1) = (alp(n+1) - h(n) * z(n)) / l(n+1)
c(n+1) = z(n+1)

do j=n, 1, -1
   cc(j) = z(j) - mu(j) * cc(j+1)
   bb(j) = (aa(j+1) - aa(j)) / h(j) - h(j) * (cc(j+1) + 2. * cc(j)) / 3.
   dd(j) = (cc(j+1) - cc(j)) / (3. * h(j))
end do

return
end

subroutine bisection(p0, p1, s, aa, bb, cc, dd, p, uinf, nu, strat, 1
   smin, cpmin, n, icon, pt)

******************************************************************************
* Date: 4/17/97
* Reference: Numerical Analysis by Burden & Faires
* Purpose: Bisection Method for Root Finding
******************************************************************************

integer i, j, n, nmax, maxiter, icon
double precision tol
parameter(nmax=200, tol=1.E-7, maxiter=1000)
double precision aa(nmax), bb(nmax), cc(nmax), dd(nmax)
double precision s(nmax+1)
double precision p0,q0,p1,q1,p,f0,fp0,f1,fp1
double precision gmin, cpmin, uinf, reo, nu, smin, strat
double precision pt

do j=1,maxiter
  do i=1,n
    if (p0.ge.s(i) .and. p0.le.s(i+1)) then
      f0=aa(i)+bb(i)*(p0-s(i))+cc(i)*(p0-s(i))**2.+dd(i)*
        (p0-s(i))**3.
      f0=f0-cpmin
      fp0=pt*(bb(i)+2.*cc(i)*(p0-s(i))+3.*dd(i)*(p0-s(i))**2.)
      gmin=((1.-cpmin)*uinf**2.)***0.5
      reo=gmin*pt*(p0-s(n/2))/nu
      q0=f0*(pt*(p0-smin)*fp0)**0.5*(1.e-6*reo)**(-0.1)-strat
    end if
  end do
  p=p0+(p1-p0)/2.
  do i=1,n
    if (p.ge.s(i) .and. p.le.s(i+1)) then
      f1=aa(i)+bb(i)*(p-s(i))+cc(i)*(p-s(i))**2.+dd(i)*
        (p-s(i))**3.
      f1=f1-cpmin
      fp1=pt*(bb(i)+2.*cc(i)*(p-s(i))+3.*dd(i)*(p-s(i))**2.)
      gmin=((1.-cpmin)*uinf**2.)***0.5
      reo=gmin*pt*(p1-s(n/2))/nu
      q1=f1*(pt*(p-smin)*fp1)**0.5*(1.e-6*reo)**(-0.1)-strat
      if (pt.eq.1.) then
        icon=i
      else
        icon=i+1
      end if
    end if
  end do
  if (abs(p1-p0)/2..lt.tol) then return endif
  if (q0*q1.gt.0.) then
    p0=p
  else
    p1=p
  end if
end do
write(*,*) 'Method Fails after',maxiter,'iterations'
return
end
The subroutine, \textit{c-p}, performs the turbulent, Couette/Poiseuille flow calculations for the flow in the gap between the vehicle and the ground plane. This calculation is used to obtain the skin friction coefficient for the gap flow and is performed once after steady state conditions are achieved for the unsteady vortex panel method calculation. This calculation uses the Reichart turbulence model.

\subsection*{E.4 \textit{c-p.f}}

\begin{verbatim}
subroutine couette(dpdx,uw,h,cf)
*****************************************************************
* Program: c-p.f
* Author: Jason Tyll
* Date: 12/26/95
* Purpose: Calculates turbulent Couette/Poiseuille flow
* using Reichart turb model
*****************************************************************
integer imax,tmax
parameter(imax=101,tmax=2000)
double precision ya,k,rho,nu,mu,uw,h,y(imax),u(imax)
double precision int(imax),tw(tmax+1), tol, diff, twf
double precision yp(imax), mut1(imax), mut2(imax), deltast
double precision dpdx,v,w,x,us,uav
integer i,j,t,iter
open(unit=1,file='c-p.out',status='unknown')
pi=4.*atan(1.)
ya=9.7
k=0.41
rho=1.177
mu=1.84e-5
nu=mu/rho
tw(1)=10.
tol=1.e-3
twf=0.

* set up y grid
do i=1,imax
  y(i)=real(i-1)/real(imax-1)
dend do

* full cosine spacing of y
do i=1,imax
  y(i)=h*0.5*(1.-cos(pi*y(i)))
dend do

* initialize deltastar (will be corrected in iterations)
\end{verbatim}
deltast=0.5*h
*
begin iteration to match Tw
do t=1,tmax
diff=0.
us=0.0
*
calculate eddy viscosity for the Reichart turbulence model
do i=1,(imax+1)/2
us=sqrt(abs(tw(t)+y(i)*dpdx)/rho)
mut1(i)=k*rho*nu*(((y(i)*us/nu)-ya*tanh(y(i)*us/nu/ya))
mut2(i)=0.192*k*rho*us*2.*h
end do
do i=(imax+1)/2+1,imax
us=sqrt(abs(tw(t)+y(i)*dpdx)/rho)
mut1(i)=k*rho*nu*((h-y(i))*us/nu-*
1 ya*tanh((h-y(i))*us/nu/ya))
mut2(i)=0.192*k*rho*us*2.*h
end do
*
calculate integrand
do i=1,imax
yp(i)=sqrt(abs(tw(t))/rho)*y(i)/nu
if (mut1(i).ge.mut2(i)) then
int(i)=dpdx*y(i)/(mu+mut2(i))
else
int(i)=dpdx*y(i)/(mu+mut1(i))
end if
end do
deltast=0.0
v=0.0
w=0.0
do j=2,imax
v=v+(int(j)+int(j-1))/2.*(y(j)-y(j-1))
if (mut1(j).ge.mut2(j)) then
w=w+mu*(1/(mu+mut2(j))+1/(mu+mut2(j-1)))/2.*(y(j)-y(j-1))
else
w=w+mu*(1/(mu+mut1(j))+1/(mu+mut1(j-1)))/2.*(y(j)-y(j-1))
end if
end do
do i=2,imax
u(i)=0.0
x=0.0
do j=2,i
u(i)=u(i)+(int(j)+int(j-1))/2.*(y(j)-y(j-1))
if (mut1(j).ge.mut2(j)) then
x=x+mu*(1/(mu+mut2(j))+1/(mu+mut2(j-1)))/2.*(y(j)-y(j-1))
else
x=x+mu*(1/(mu+mut1(j))+1/(mu+mut1(j-1)))/2.*(y(j)-y(j-1))
end if
end do
u(i)=u(i)+uw*x/w-x/w*v
*
correct deltastar
APPENDIX E. COMPUTER CODES

```
deltast=deltast+(2.-(u(i)+u(i-1))/uw)/2.*(y(i)-y(i-1))
end do
uav=0.0
do i=1,imax
  uav=uav+u(i)
end do
uav=uav/real(imax)

diff=abs(tw(t)-mu*(uw-v)/w)
if (diff.le.tol) then
  twf=tw(t)
goto 10
end if

* wall shear corrected
  tw(t+1)=mu*(uw-v)/w
end do

write(1,*) 'No Convergence'
10 continue

* output
write(1,*) twf, twf/0.5/rho/(uw*2.)**2., twf/0.5/rho/(uw)**2.
write(1,*) deltast, uav
do i=1,imax
  write(1,*) y(i)/h, u(i)/uw
end do

cf=mu*abs((u(imax)-u(imax-1))/(y(imax)-y(imax-1)))/(0.5/rho/(uw)**2.
close(1)
return
end
```
The file, declare, is a collection of “parameter” statements used for the weight and cost models. The parameters include information about the mission details and the assumed economic factors.

E.5  declare.h

******************************************************************************** Program: declare.h
* Author : Mark Eaglesham
* Date :
* purpose: SETUP COMMON INPUT VARIABLES FOR MAGLEV MASS,
* MANUFACTURING COST, LIFE CYCLE, & OPERATING COST
* PROGRAMS.
* Use "include" statement in each subroutine.
********************************************************************************
C DECLARE FOR
C SETUP COMMON INPUT VARIABLES FOR MAGLEV MASS, MANUFACTURING
C COST & LIFE CYCLE OPERATING COST PROGRAMS

C INITIALIZE PARAMETER NAMES
double precision MPERMO,MCONT,MSEATS,MANC,MFUSE,LIFTMO,
x LIPPOL,MPOLE,
x POWPOL,AUPOLR,NPASS,MAVEP,MLUGG,SPEED,TRDIST,ADENS,
x FRAREA,CD,CL,MNOSE,MCONTROL,MSEAT
double precision NUMMOT,MMOT,MPASS,MVEHIC,DRAG,LIFT,TRTIME,
x TOTTPW,TERGY,VMC,VM,
x ENCOT,COSTPP,COSTPM,NTRIPS,INTEREST,PVFACTOR,INFLATN,
x ILEVIT,PLEVIT,CLC
DOUBLE PRECISION X,C1,NZ,NMODULE,L,SF,D,X0,WREST,ROOT,F,DFDX,
x DF,DX,DFMAX,EPS,MULT
DOUBLE PRECISION ELECHA,ELEDEM,PLOAD,OPHOURS,YEAR,PRPASS
DOUBLE PRECISION GROWTH,CGUIDPM,NVEHIC,QUANTITY,MTV,RE,RT
DOUBLE PRECISION NQ,RL,LMOD,WMOT,WCONTROL,WPASS
DOUBLE PRECISION NUMMOTOR,MVEHIC,WDG,WFUSE,WEMPTY,Q
DOUBLE PRECISION PDRAG,FVEHIC,FLEVIT,PL,VL,FL,PAUX,
DOUBLE PRECISION PRPASSKM,NTICKETS,REVENUE,TCGUID,CINVEST
DOUBLE PRECISION TRIPS,COOPERATE,CASHFLOW,DEPN,TAXINC,TAXPAID
DOUBLE PRECISION ATCF,PVATCF
DOUBLE PRECISION ENGH,ENGTC,TOOLHC,TOOLMC,MFGH,MF,GC,QCH,QCC,
x DEVG,TESTC,MAITLC,CVEHIC,CFLEET
INTEGER IMAX,N
INTEGER I, LIFE

C INPUT PARAMETER VALUES
PARAMETER(MPERMO=52., MCONT=47200., MSEAT=14.55, MANC=0.
x, NMODULE=1., VCM=1., VMM=1.)
C Mass/Motor:MPERMO kg/unit: 1470 kg given lit.ref:ref., value
C Mass of Controls/Module:MCONT kg: assumed from sum of
C other values
C Mass of Furnishings:MSEAT kg: seat 14.55kg (Raymer)* 100 =
C 1455kg
APPENDIX E. COMPUTER CODES

C Mass of Ancillary Equipment: MANC kg: assumed included in MCONT
C Number of Vehicle Modules: NMODULE 1, 2, 3, 4, DEPENDING ON CONFIGURATION
C Vehicle Cost Modifier: VCM=1 aluminum, =1.1 composite
C Vehicle Mass Modifier: VMM=1 aluminum, =0.8 composite

PARAMETER(AUPOWR=85., EFF=0.82)
C Auxillary power consumption: AUPOWR 85[kW].
C GMSA final report: 49-50
C Converter Station Output Efficiency: EFF 0.82
C Efficiency at LSM is approx 1.00

PARAMETER(MAVEP=93.2, MLUGG=0., SPEED=134., TRDIST=800.)
C Ave.mass.of passenger: MAVEP 93.2 [kg]
C INCL. LUGGAGE. Shaw, Grumman
C Mass of Luggage: MLUGG 0 [kg] included with MAVEP.
C Shaw, Grumman
C Speed of Vehicle: SPEED 134 [m/s] GMSA final report
C Trip distance: TRDIST 800 [km] GMSA final report

PARAMETER(ADENS=1.177, FRAREA=3.5)
C density of air: ADENS 1.177 [kg/m3] given
C frontal area:S=FRAREA 3.5*1=3.5 sq.m
C given lit.ref:3.8m w x 3.6m len(ref Allen & Ghalli, 1993)

PARAMETER(ELECHA=0.05, ELEDEM=7.50, NTRIPS=19200., PLOAD=2000.,
C xOPHOURS=16., YEAR=365., LIFE=15, PRPASS=0., GROWTH=0.04,
C xCGUIDPM=19800000., NVEHIC=100.)
C Energy Charges: ELECHA=$ 0.05 [$/kW.h]
C given lit.ref: ref., value
C Demand charge: ELEDEM=$ 7.50 [$/mo per kW]
C given lit.ref: ref. value
C Number of trips per month: NTRIPS=2000/50*16*30=19200
C Passenger load: PLOAD=2000 per hr
C Operational hours: OPHOURS=16 hrs/day
C Service level: YEAR=365 days/year
C Expected lifetime: LIFE=15 years
C Price per passenger: PRPASS=0 i.e. no revenue (in 1995)
C Ave.growth in traffic: GROWTH=4% per annum
C Guideway cost/mile: CGUIDPM= $19,800,000 per mile (Deutch)
C Vehicle size: 50 passengers
C Vehicle availability: 99% of operational time NOT USED
C No. vehicles reqd.: 100 vehicles: NVEHIC=100

PARAMETER(INTEREST=1.06, TAX=0.0, INFLATN=0.03)
C Interest rate: INTEREST=6% per annum
C Depn: sum of years digits over vehicle life
C Tax rate: TAX=0.0 or 50%
C Average inflation: INFLATN=3% per annum

PARAMETER (QUANTITY=100., MTV=2., RE=59.10, RT=60.70, RQ=55.40,
APPENDIX E. COMPUTER CODES

x

C Quantity of vehicles to be built: QUANTITY = 100
C Maglev Test Vehicles: MTV = 2
C Labor Cost Rate: Engineering: RE = 59.10
C Labor Cost Rate: Tooling: RT = 60.70
C Labor Cost Rate: Quality Control: RQ = 55.40
C Labor Cost Rate: Manufacturing: RM = 50.10

PARAMETER (NZ = 3.00, D = 11.6, LMOD = 79.2)
C Nz: ultimate load factor = 1.74
C L: fuselage struct. length = 71.61 ft (REMOVED L = 71.61,)
C D: fuselage struct. depth = 12.54 ft
The subroutine, `mvehicc`, calculates the vehicle mass and the empty weight using the geometry definition as input.

### E.6 mvehicc.f

```fortran
subroutine mvehicc(MVEHIC,WEMPTY)

integer nmax
parameter(nmax=104)
double precision xx(nmax+1),yy(nmax+1)
open(unit=9,file='mag.dat',status='old')

C Number of Motors: NUMMOT [#]=24 per module , GMSA final report.
NUMMOT=24.*NMODULE

C Mass of Motors: MMOT [kg]
MMOT=NUMMOT*MPERMOC

C Mass of Controls: MCONTROL: [kg]
MCONTROL=MCONT*NMODULE

C Number of Passengers: NPASS
NPASS=NMODULE*50.

C Mass of Passengers: MPASS 9506.4 [kg] calc. product
MPASS=NPASS*(MAVEP+MLUGG)

C Mass of Seats
MSEATS=NPASS*MSEAT

C CONVERSION OF MASSES TO WEIGHTS: kg TO lbf
WMOT=2.205*MMOT
WCONTROL=2.205*MCONTROL
WPASS=2.205*MPASS
WSEATS=2.205*MSEATS
WREST=WMOT+WCONTROL+WPASS+WSEATS

C GEOMETRIC PARAMETERS OF VEHICLE
L=NMODULE*LMOD

calculate SF (surface area)
do I=1,nmax+1
   read(9,*) xx(I),yy(I)
end do
SF=0.
do I=2,nmax+1
   SF=SF+sqrt((xx(I)-xx(I-1))**2.+
(yy(I)-yy(I-1))**2.)*D
not unit width for the case of surface area (so we can calc
reasonable weight
end do
do I=2,nmax+1
   SF=SF+0.5*(xx(I)-xx(I-1))*(yy(I)+yy(I-1))*2.
end do

C convert units from m^2 to ft^2
SF=SF*3.3**2.

C PROGRAM TO SOLVE FOR WDG USING
C WDG=WREST+WFUSE*NMODULE
C WFUSE=0.328*(WDG*NZ)**0.5*(L)**0.25*(SF)**0.302*(L/D)**0.1
C1=0.3280*NZ**0.5*L**0.25*SF**0.302*(L/D)**0.1
X0=1300000.
EPS=0.1
IMAX=100
DXMAX=1.0E8
MULT=1.
CALL NEWTON(X0,C1,WREST,EPS,IMAX,DXMAX,MULT,WFUSE,
   XF,DFDX,N)

WDG=WFUSE*NMODULE+WREST
WEMPTY=WDG-WPASS

C Convert WFUSE to kg
MFUSE=WFUSE/2.205

C Mass of Vehicle: MVEHIC=61369.4 [kg] calc.sum
C (Lever- for 2 mod vehicle)
MVEHIC=MMOT+MCONTROL+MPASS+MSEATS+MFUSE*NMODULE
C Vehicle mass modifier: VMM input as parameter
MVEHIC=VMM*MVEHIC

C SUBROUTINE TO CALCULATE ROOT OF 2 EQUATIONS USING NEWTON'S METHOD
SUBROUTINE NEWTON(X,C1,WREST,EPS,IMAX,DXMAX,MULT,
   X ROOT,F,DFDX,N)

VARIABLE DECLARATIONS
INCLUDE "declare.h"

ITERATIONS
DO 1 I=1,IMAX
   DFDX=1.-0.5*C1*(NMODULE*X+WREST)**(-0.5)*NMODULE
   F=X-C1*(NMODULE*X+WREST)**0.5
   DF=DFDX
   IF(DF.EQ.0.)THEN
      WRITE(*,112)I,X,F
      WRITE(*,*) 'ERROR IN NEWTON'
   END IF
1 CONTINUE
RETURN
RETURN
ENDIF
DX=-MULT*F/DF
IF(ABS(DX).LT.EPS)THEN
  ROOT=X+DX
  N=I
  RETURN
ELSEIF(ABS(DX).GT.DXMAX)THEN
  WRITE(*,110)I,DX
  WRITE(*,*) 'ERROR IN NEWTON,METHOD DIVERGING'
  RETURN
ENDIF
X=X+DX
1 CONTINUE
WRITE(*,111)X,F,DX,I
WRITE(*,*) 'ERROR IN NEWTON,METHOD NOT CONVERGING'
C MESSAGES
110 FORMAT('In iteration',I6,'dx=',F15.1,'larger than limit')
111 FORMAT('Excessive iterations after max steps x:f(',F15.1,')=',F15.1/'latest dx=',F15.1,'ITERATIONS=',I6)
112 FORMAT('In iteration',I6,'df=0: problem with df/dx',2F15.4)
RETURN
END
The subroutine, \textit{opcost}, calculates the direct operating cost for a single trip using vehicle mass, aerodynamic coefficients, and system performance parameters as input. The power requirements are calculated and the cost is determined using electricity cost and demand charges.

\textbf{E.7 opcost.f}

```
subroutine opcost(CD,CL,MVEHIC,FLEVIT,ENCOST)
*****************************************************************
** Program: opcost.f
** Author : Mark Eaglesham
** Date : 
** purpose: Calculates the direct operating cost for single trip.
** Calculates power requirements.**
****************************************************************
INCLUDE "declare.h"
C Dynamic pressure of freestream air: \( q = \frac{1}{2}.\rho.V^2 \) (RAYMER p.260)
Q=0.5*ADENS*SPEED**2.
C Aerodynamic Drag Force \( D = q.S.Cd \) [kN] (Raymer p.260)
DRAG=Q*FRAREA*CD/1000.
C Lift: \( L = q.S.Cl \) [kN] (Raymer p.260)
LIFT=Q*FRAREA*CL/1000.
C Time for trip: TRTIME =7939s =132.3 min = 2.2053hr.
C given (in hrs)
TRTIME=TRDIST/SPEED*1000./3600.
C Power consumption for DRAG: \[kW\]
PDRAG=DRAG*SPEED/EFF
C Force downward \[kN\] due to vehicle mass plus safety
C factor of 1.5
FVEHIC=MVEHIC*9.81./1000.
C Force required for magnetic levitation \[kN\]
FLEVIT=FVEHIC-LIFT
C Calculate force normal to magnets: FLEVIT/COS(35)
PI=4.*ATAN(1.)
FLEVIT=FLEVIT/COS(35.*PI/180.)
C Read off polynomial curve for current use[kA]
C [GMSA final report]
ILEVIT=7.E-9*FLEVIT**3.-2.E-5*FLEVIT**2.+0.0526*FLEVIT+10.045
C Initial method: Levitation power consumpt: POWLIF = 2100.4 \[kW\]
C calc.MAE
C POWLIF=2100.4
C Power consumption for Levitation and guidance: \( P = V*I \) \[kW\]
VLEVIT=42.
PLEVIT=VLEVIT*ILEVIT
```
C Auxillary power consumption: PAUX [kW]
PAUX=AUPOWR
C Total power consumption rate [kW]
TOTPOW=PDRAG+PLEVIT+PAUX
write(*,*) 'PDRAG=',PDRAG, 'PLEVIT=',PLEVIT
C ENERGY CONSUMED BY VEHICLE OVER TRIP [kW.h]
TENERGY=TOTPOW*TRTIME
C ENERGY COST CALCULATION
C Energy Cost: [$] : calc.sum
ENCOST=(TENERGY*ELECHA)+(TOTPOW*PLOAD/(NMODULE*50.)/
1 NTRIPS*ELEDEM)
C Total cost per passenger per trip: COSTPP 0.0025 [$/passenger]
NPASS=NMODULE*50.
COSTPP=ENCOST/NPASS
C Cost per passenger km ($/passenger km)
COSPPM=COSTPP/TRDIST
RETURN
END
The subroutine *lifecost*, calculates the acquisition cost, yearly operating cost, and life cycle cost using a discounted cash flow analysis. The required input to this model is the empty weight, energy cost for a single trip, and economic factors.

### E.8 *lifecost.f*

```fortran
subroutine lifecost(WEMPTY,ENCOST,CLC,COPERATE,CVEHIC,
                     CINVEST)
****************************************************************
* Program: lifecost.f
* Author : Mark Eaglesham
* Date : 
* purpose: Calculates acquisition cost and yearly operating cost.
*          Performs discounted cash flow analysis to obtain net
*          present value.
****************************************************************
INCLUDE 'declare.h'

C Price per pass. km:
PRPASSKM=PRPASS/TRDIST

C Tickets sold: 23,360,000 in year zero
NTICKETS=PLOAD*OPHOURS*YEAR

C Revenue per year: $350,400,000
REVENUE=NTICKETS*PRPASS

C Cost of Infrastructure
C Total guideway cost: $9,127,800,000
TCGUID=CGUIDPM*TRDIST/1.609

C COST OF VEHICLES (DACPA IV COST MODEL: RAYMER P.498)
V=1.943*SPEED
ENGH=4.86*WEMPTY**0.777*V**0.894*NVEHIC**0.163
ENC=RE*ENGH
TOOLH=5.99*WEMPTY**0.777*V**0.696*NVEHIC**0.263
TOOLC=RT*TOOLH
MFGH=7.37*WEMPTY**0.82*V**0.484*NVEHIC**0.641
MFHG=RM*MFHG
QCH=0.133*MFHG
QCC=Q0*QCH
DEV=45.42*WEMPTY**0.630*V**1.3
TESTC=1243.03*WEMPTY**0.325*V**0.822*MTC**1.21
MATLC=11.0*WEMPTY**0.921*V**0.621*NVEHIC**0.799
CFLEET=ENGC+TOOLC+MFGC+QCC+DEV+TESTC+MATLC

C Vehicle cost modifier: Material=composite=>VCM=0.8 TO 1.1
CFLEET=VCM*CFLEET

C Fleet cost: $825,000,000 baseline
CVEHIC=CFLEET/NVEHIC

C Investment Cost
CINVEST=CFLEET

C Number of trips made per year
NPASS=NMODULE*50.
```

```
TRIPS = PLOAD / NPASS * OPHOURS * YEAR
C Operating Cost
C OPERATE = -ENCOST * TRIPS
C Initial Cash Flow = -CINVEST
CLC = -CINVEST
C Annualized net present value of cash flows for 15 year life
DO I = 1, LIFE
   ATCF = OPERATE
   PVFACTOR = INTEREST ** REAL(I)
   PVATCF = ATCF / PVFACTOR
   CLC = CLC + PVATCF
   OPERATE = (1. + GROWTH) * (1. + INFLATN) * OPERATE
END DO

RETURN
END