A 3-D TUNNEL CORRECTION PANEL METHOD FOR SWEPT TAPERED AIRFOILS WITH SEPARATION

by

John Henry Burns

B.S.A.E., University of Colorado Boulder
(1985)

SUBMITTED TO THE DEPARTMENT OF
AERONAUTICS AND ASTRONAUTICS
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MASTER OF SCIENCE IN
AERONAUTICS AND ASTRONAUTICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 1988

Signature of Author _____
Department of Aeronautical and Astronautical Engineering
January 29, 1988

Certified by _____
Dr. Judson R. Baron
Thesis Supervisor

Accepted by _____
Dr. Harold Y. Wachman
Chairman, Departmental Graduate Committee

A 3-D TUNNEL CORRECTION PANEL METHOD FOR SWEPT TAPERED AIRFOILS WITH SEPARATION

by

JOHN HENRY BURNS

Submitted to the Department of Aeronautics and Astronautics
on January 29, 1988 in partial fulfillment of the requirements
for the degree of
Master of Science in Aeronautics and Astronautics

ABSTRACT

Wind tunnel walls, by their presence, alter air flow around the model and thus the data gathered. Classical methods of wind tunnel wall corrections are of questionable accuracy for extreme test conditions (i.e. high angle of attack with separation). A computational method has been developed to calculate wind tunnel wall interference for wings of arbitrary sweep and taper. Additional wings may be present to represent a horizontal tail, canard, or multiple wings. The closed test section tunnel is of arbitrary eccentricity and a vertical wing with a ground board can be simulated. A three dimensional vortex panel method is used to model the wing and its wake. A two dimensional boundary layer analysis is used to find the separation locus. The separation locus and wake shape are found by iteration.

The results show proper convergence trends for a range of sweep, dihedral, taper, and aspect ratio. The data shows proper behavior for variations of the wing planform and tunnel eccentricity.

Thesis Supervisor: Dr. Judson R. Baron
Title: Professor of Aeronautics and Astronautics

To my Family.

Acknowledgements

Thanks to all of my friends who have helped make my stay at M.I.T. both productive and pleasant, with special thanks to Jon Mishler. Thank you, Tim Gilbert, Susan Tucker, Dave Kikorian, Don Robinson, and Octavio Marenzi for your help with the figures.

At the Wright Brothers Facility I have had the pleasure of working with many fine graduate students, all of whom have influenced myself and my work: John Busch, Charles Boccadoro, Tim Gilbert (my favorite civil engineer), Don Robinson, and Mike D'Angelo.

I wish to thank Frank Durgin, assistant director of the Wright Brothers Facility, for his insights into all aspects of wind tunnel testing.

I wish to express my sincere appreciation to Professor Judson R. Baron, my advisor, for his continuous assistance and availability during my stay at M.I.T.

Table of Contents

List of Figures

NOMENCLATURE

[A]	Coefficient matrix
AR	Wing aspect ratio
B	Tunnel semi-width
{B}	Right hand side column matrix
b	Wing span
b'	Panel span
c	Wing root chord
c'	Panel chord
Cl	Coefficient of lift/unit span
CL	Coefficient of lift
CLfa	Coefficient of lift in free air
CLt	Coefficient of lift in tunnel
F1,F2,G1,G2	Induced velocity subfunctions
H	Boundary layer shape parameter
Ht	Tunnel semi-height
inc	ith chordwise panel
ins	ith spanwise panel
nc	Number panels along chord
ns	Number of panels across semi-span
p	Static pressure
P	Total pressure
Re	Reynolds number based on wing root chord
Re_x	Reynolds number based on distance x

Re_θ	Reynolds number based on boundary layer momentum thickness
$s = b/2$	Wing semi-span
S	Wing area
u,v,w	Velocity components in global frame
u',v',w'	Velocity components in panel frame
[u],[v],[w]	Induced velocity matrices - global frame
[u'],[v'],[w']	Induced velocity matrices - local panel frame
$\vec{V}$	Velocity vector in global coordinates
$\vec{V}'$	Velocity vector in panel coordinates
$\vec{V}_\infty$	Free stream velocity
WL	Wake length
{X}	Column matrix of panel strengths
x,y,z	Global coordinates
x',y',z'	Panel coordinates
α	Angle of incidence
χ	Dihedral angle
γ	Circulation / unit length
Γ	Circulation
$\Lambda_{1/4}$	Sweep angle of wing 1/4 chord
Λ	Panel sweep angle
λ	Tangent of panel sweep angle
Φ	Velocity potential
ϕ	Perturbation velocity potential
θ	Boundary layer momentum thickness
μ	Coefficient of dynamic viscosity
ν	Coefficient of kinematic viscosity

τ Shear stress

CHAPTER 1

INTRODUCTION

The historic flight of the Wright brothers in 1903 demonstrated the value of wind tunnel testing, which was used methodically to design their airfoils. The fact that Rockwell spent 46,500 hours testing models of the space shuttle in various wind tunnels during the design phase of that program [Whitnah and Hillje (1)] shows the ever continuing need for wind tunnel testing. It is still impractical to solve the full Navier-Stokes equations for a flight vehicle either analytically or numerically. Analytical solutions exist only for very special boundary conditions and the computational effort needed for the direct numerical modeling of turbulent shear layers at flight Reynolds numbers is beyond the ability of any computing machines for the forseeable future. Wind tunnel testing remains crucial for the successful design and optimization of flight vehicles due to its ability to directly model the physics without simplifying assumptions, while encompassing all scales of motion and time.

While the solution by direct analogy is complete, it is only as accurate as the analogy. The similarity conditions for a low speed test in which gravitational forces and flexibility are negligible requires the matching of model

geometry, Mach number, and Reynolds number. The matching of both Mach number and Reynolds number is difficult to achieve unless either the model is full-sized or a custom working fluid is used in the tunnel. If sea level air is used in a tunnel the largest model possible allows for the closest match of both Reynolds number and Mach number.

While the physics of a wind tunnel test may be correct, the boundary conditions match only in the limit of an infinitely large tunnel diameter compared to a representative length of the model. The boundary conditions for free flight are a uniform flow field at infinity, which is equivalent to an unaltered streamtube at infinity, but not to the tunnel conditions of an unaltered streamtube at the tunnel walls. The effects of such flow field constraints is collectively known as wind tunnel wall interference. The wall interference alters the aerodynamic behavior of the model compared to free flight conditions. Thus, the effects of the wall interference must be accounted for before the data is representative of free air conditions.

If an exact calculation of the wall interference followed from solving the complete fluid governing equations for a model both in free air and in the tunnel, the ability to completely solve the governing equations would limit the need for wind tunnel testing. However, wind tunnel verification

would still be necessary to have the confidence to build a prototype. The wind tunnel analogy remains the only "method" for solving the flow field completely.

Classical methods of wind tunnel wall corrections approximate the wing, wake, and trailing vorticity using flow singularities. Images of the singularity distribution representing the wing, wake, and trailing vorticity are arranged in a configuration such that the wind tunnel wall boundary conditions are implicitly enforced. The interference flow field of the tunnel is represented by the image singularities. This interference can be interpreted as a variation in the flow direction, speed, and curvature. However, in the classical approach, at high angles of attack, wake curvature and flow separation are ignored and therefore are questionable under such conditions. Rae and Pope (2) present a detailed description of the effects of wall interference and classical correction methods. Joppa (3) provides a short history of the development of wall corrections including Prandtl and Glauert's work. Classical wall corrections are linear because the strength of the image vortices are directly proportional to the lift of the model.

The major effects of wall interference on a typical wing flow for a closed wind tunnel are summarized as:

1) An acceleration of the flow around the model known as "solid blockage" is due to the volume of the model and the constraint of the walls.

2) An acceleration of the flow around the low speed wake known as "wake blockage" is due to the conservation of mass flow within the tunnel walls

3) An alteration of the free air spanwise variation of angle of attack significant for wings that span more than 80% of the tunnel, causing the tips to stall earlier than in free air.

4) A change of flow curvature about the wing increasing the lift and moment.

5) The normal downwash at the wing is changed, causing an increased lift and decreased drag.

More recently other methods of wind tunnel wall interference corrections have been developed which use computers to carry out calculations too long and tedious to be reliably calculated by hand in a reasonable amount of time. Methods of calculating the wall interference flow based on a "wall pressure signature" (4,5) require the use of extensive wall pressure data. They are similar to higher order classical techniques (with the additional information of the pressures at the walls) in that the interference flow of the tunnel is calculated and utilized to calculate the corrections.

The most elegant approach attempts to completely remove the wind tunnel wall interference instead of correcting for it. The adaptive wall method (6,7,8) alters the streamtube defined by the wind tunnel walls by matching it to the

corresponding streamtube in free air. Practical limits in the adjustments may prevent the complete elimination of such interference, but the technique is used to minimize wall interference to some extent. The method requires extensive pressure sensing and the iterative method of adjusting the walls requires solving for the pressures of the free air streamtube. The expense of an automated system that adjusts the wind tunnel walls prevents this idea from being applicable to most tunnels.

Hess and Smith (9) developed a pioneering panel method for calculating the potential flow around arbitrary bodies, which was published in 1966. Since then many investigaters (10,11,12,13) have introduced other panel methods to calculate the flow over a wing. The wing is represented by distributed panels, where each panel is a singularity with a strength determined such that the tangential flow condition and Kutta condition are enforced. A natural extension of classical methods of wall corrections, given the digital computer to carry out the calculations, is the use of multiple singularities to model the wing in greater detail, and either the use of images to implicitly model the walls or the use of singularities to directly model the tunnel surfaces. Thus, vortex lattice (14) and panel (15,16,17) methods for wall corrections are the descendants of classical methods.

Olechowski (14) developed a vortex lattice method for calculating wall interference including the effects of separation. The method requires the separation locus of the wing to be determined in the wind tunnel test. The separation locus for the free air calculation is assumed to be the same as the separation locus in the tunnel. This assumption is questionable because the wall interference may affect the separation locus on the wing.

Tennison (15) developed a distributed vorticity panel method which included a boundary layer analysis routine to find the separation locus. The fact that the wall interference may alter the separation locus of the model compared to free flight conditions is accounted for by his approach. However, the modeling used an open wake with straight parallel boundaries.

Both Olechowski and Tennison developed their methods at the Wright Brothers Facility at M.I.T. The predominance of complicated wing configuration designs makes those methods which are limited to a single rectangular wing without dihedral unapplicable to most tests. At present the only aircraft with wings that are close to retangular planform are small private airplanes such as a Piper Cub, and even those planes typically have a small amount of sweep, taper and dihedral. Thus, the features needed include the use of

multiple wings of arbitrary sweep, taper and dihedral. This work was undertaken at the Wright Brother Facility to provide a design method which would be applicable to wind tunnel tests of wings that are somewhat more represenative of "real" airplanes.

The following is a formulation of a 3-D wall interference method that uses vortex panels to represent the model, walls, and free shear layers. Spanwise stations are treated with a 2-D boundary layer analysis to determine a separation location for each strip of the wing. The wake is adjusted independently at each spanwise station until the upper and lower edges of the wake lie along streamlines. There is no constraint on the shape of either the upper or lower wake. On the final iteration both the separation locus on the wing and the wake shape have converged. The tunnel is excluded when solving for the model in free air. This is equivalent to placing the model in a tunnel of infinite diameter compared to a representative length of the model. The effect of the displacement thickness of the boundary layers is not modeled at this time. The wake consists of unconnected strips of the same dihedral as the wing, and the rollup of the trailing vortices is not modeled.

CHAPTER 2

QUALITATIVE DESCRIPTION AND MODELING OF FLOW OVER A WING

Figure 2-1 illustrates the physical problem of separated flow. The ideal modeling is described later in this chapter and is obtained from the technique explained more fully in Chapter 4. Before examining the separated case shown in Figure 2-1 a brief explanation of the attached case is in order.

The boundary layers in the so-called attached case in fact effectively separate from the airfoil at the upper and lower trailing edges, resulting in an absence of a separated region over the airfoil itself, and confining the viscous wake region to the combined thickness of the trailing edge and both boundary layers. From Helmholtz's vortex theorem a two dimensional wing creates no new net vorticity in a steady flow. Therefore, the boundary layer vorticity that is shed from the upper and lower trailing edges must be equal in magnitude and opposite in orientation. This corresponds to a Kutta condition of no net vorticity at the trailing edge. The 3 dimensional case requires a modification of the Kutta condition modeling for attached flow. This modeling problem, its solution, and the underlying cause are discussed in Chapter 3.

The comparative advantage of panel methods over classical methods as a basis for wall interference effects originates from an increased accuracy of the wing modeling, and would be lost if the separation was not included when present. There are three distinct types of separation: bubble, leading edge, and trailing edge. Bubble separation usually occurs when a laminar boundary layer separates from the surface of the airfoil. The resulting thin free shear layer changes from laminar to turbulent, due to the instability of the thin free shear layer, and the turbulent thin free shear layer reattaches to the airfoil as a turbulent boundary layer. The latter is robust because the turbulent mixing brings high energy fluid into the boundary layer, allowing it to withstand a relatively greater adverse pressure gradient before separating than does a laminar boundary layer. Bubble separation is divided into long and short categories depending upon the length of the separated region. In the present modeling, separation is assumed to be of the short bubble variety for laminar situations. The separation bubble is modeled as being of negligible length and any thickening of the boundary layer is neglected. When the angle of attack is gradually increased the boundary layer eventually leaves the airfoil surface and jumps suddenly to near the leading edge. In the case of trailing edge separation, the point of separation moves smoothly forward from the upper trailing edge as the angle of attack increases past that angle for which

attached flow can be maintained. The maximum angle of fully attached flow varies with the wing section, Reynolds number, surface roughness, and free stream turbulence. An angle of approximately 8 degrees (18) represents the maximum angle of fully attached flow for a NACA 0012 airfoil at a Reynolds number between 3 and 9 million. This airfoil is subject to the trailing edge type of separation.

The steady separated flow over a wing at high Reynolds numbers and low Mach numbers consists of four distinct regions as shown in Figure 2-1. An examination of each region and their interactions suggests which physically reasonable assumptions are needed to make the problem tractable, and also suggests a technique for modeling the separated flow over a wing.

Initially irrotational upstream of the airfoil, the flow remains irrotational in the essentially potential flow region (exterior to all shear layers and wakes) because everywhere in that region the shear is essentially negligible. The flow in such a potential region interacts with its surroundings only through pressure forces, and has been modeled here by a Laplace equation which assumes it is both irrotational and incompressible (low Mach number).

The boundary layer is the thin region between the solid

surface of the airfoil and the outer potential flow. The shear at the inner solid boundary generates large viscous forces and vorticity which is confined to the boundary layer. The displacement thickness of the boundary layer has not been added to the thickness of the wing. A number of investigators (19) have developed methods that accurately predict the behavior of boundary layer parameters from the pressure distribution along the layer's outer boundary adjacent to the potential flow region. The boundary layer has been modeled here using Thwaites' (20) laminar analysis, a modification of Michel's transition criteria (19) and Stratford's (22) turbulent separation criteria.

A thin free shear layer, if present, is fed by a separating boundary layer and defines the boundary between essentially two inviscid regions, such as for example, an outer potential flow and an inner separated region. Shear and viscous forces are relatively moderate in the thin shear layer when compared with those in a boundary layer. The thin free shear layer is modeled as a discontinuity surface by a continuation of the surface vortex panels from which it extends. At the trailing edge of an airfoil with upper surface separation the boundary layer from the lower surface leaves to form a second thin free shear layer. The thin free shear layers then follow the streamlines of the flow and are essentially slip surfaces across which a jump in tangential

velocity exists. The vortex strength of the panels is constant for the thin free shear layers and the vorticity of the two sheets is of opposite orientation.

The upper wing surface downstream of the separation point and both thin free shear layers define the boundaries of the separated region. The latter are characterized by low vorticity, negligible viscous forces, and a constant but lower total pressure than the potential flow region, due to the entropy production in the shear layers. The separated region is presently modeled as a potential flow. The momentum deficit in a real fluid extends beyond the separated region.

CHAPTER 3

POTENTIAL FLOW EQUATION

The potential flow about the wing is assumed to be steady, 3-dimensional, incompressible, and inviscid. Viscous phenomena are confined to regions of infinitesimal thickness along the wing and wake boundaries and introduced solely as a mechanism for triggering separation. Irrotationality for a velocity field $\vec{V}$ can be expressed as:

$$\text{curl } \vec{V} = 0 \qquad [3\text{-}1]$$

For any scalar function Φ the following identity (23) exists:

$$\text{curl grad } \Phi = 0 \qquad [3\text{-}2]$$

Therefore, a potential function Φ exists for any irrotational velocity field $\vec{V}$ and irrotationality is a necessary condition for it to exist. The nature of irrotationality ensures that the cross derivatives of the potential function are independent of the order of differentiation.

$$\vec{V} = \text{grad } \Phi \qquad [3\text{-}3]$$

Also, three dimensional flow incompressibility can be expressed as:

$$\text{div } \vec{V} = 0 \qquad [3\text{-}4]$$

Substituting Equation 3-3 into 3-4 gives the Laplace equation.

$$\text{div grad } \Phi = 0 \qquad [3\text{-}5]$$

This linear partial differential equation displays two important properties applicable to the solution of potential flow problems. First, the equation is linear; in accordance with the superposition principle any number of velocity fields or potentials that satisfy it can be combined to form a new solution. Second, the flow can be expressed in terms of the velocity potential Φ instead of $\vec{V}$, replacing a vector problem with a scalar problem.

There are a number of methods suited to solving Laplace's equation. However, for the problem at hand - a wing in a potential flow field - panel methods dominate due to their computational efficiency. They solve the entire flow field by specifying a singularity distribution on the boundary that produces it. Once the singularity strengths are determined, when using a panel method with N panels, the extra effort

needed to determine the velocity at each additional point is of order (N) since the influence coefficients of every panel must be calculated for each additional point. While some methods solve for velocities over the entire flow field at the same time, their computational requirements are generally much larger for a solution of the same accuracy. A panel method was chosen for the relative efficiency; the solution of the complete flow field is achieved in order to obtain the velocity distribution over the wing without obtaining details of the complete velocity field. The benefits of a vortex panel method include intuitive modeling of the flow physics, ease of separation modeling, simplicity, and the need for fewer total panels than methods that use both source and vortex panels.

THEORETICAL FOUNDATION OF PANEL METHODS

The theoretical justification of panel methods is derived using Green's third identity (24). The potential flow function Φ meets the requirement of continuity in the domain for Green's third identity to apply. Using the superposition principal the potential Φ is split into two components: a constant velocity potential for the free stream at infinity, and a perturbation velocity potential, which is induced by a singularity distribution on the boundaries of the potential

flow region. The split is:

$$\vec{V} = \vec{V}_\infty + \vec{v} = \text{grad } \Phi = \vec{V}_\infty + \text{grad } \phi \qquad [3\text{-}6]$$

Green's third identity (Equation 3-7) applied to the perturbation velocity potential ϕ determines the value of ϕ at any point P from the Laplacian of ϕ, and the value of ϕ and its normal derivative directed into the potential region on the boundary containing the point P. Green's third identity is also applicable to the freestream flow which can be thought of as the result of a distribution of singularities at infinity. However, the origin of the free stream velocity does not affect the perturbation velocity potential calculation.

$$\phi(P) = -\frac{1}{4\pi}\iiint_\Omega \frac{1}{r_{PQ}}\,\Delta_Q \phi$$

$$-\frac{1}{4\pi}\oiint_{\partial\Omega} \frac{1}{r_{PQ}}\,\frac{\partial \phi}{\partial n_Q}$$

$$+\frac{1}{4\pi}\oiint_{\partial\Omega} \frac{\partial}{\partial n_Q}\left(\frac{1}{r_{PQ}}\right) \qquad [3\text{-}7]$$

where the subscripts indicate

Ω - the region

$\partial\Omega$ - the boundary of the region

PQ - from P to Q

Q - at point Q

The first term on the right hand side vanishes (Equation 3-5). The second and third terms on the right hand side represent source and vortex distributions respectively, on the boundary of the potential flow region. The source distribution is responsible for any discontinuities of the flow velocity in the normal direction at the boundary. If the internal perturbation velocity field is chosen such that the velocity in the normal direction is continuous at the boundary, then the second term on the right hand side of Equation 3-7 is identically zero. Similarly, for the tangential flow velocity to be continuous across the boundary, the vortex distribution must be identially zero. The use of both source and vortex distributions on the boundary allows independent boundary conditions to be enforced on both sides of the singularity distribution. Since the potential flow field internal to the wing can be arbitrarily chosen without affecting the potential flow field outside of the wing, either the source or vortex distribution on the boundary can be set to zero when modeling a nonlifting body by adjusting the boundary conditions for the flow internal to the body. The case of a lifting body such as a wing requires the use of vortex singularities since a source distribution cannot induce

circulation. Note that the condition of reducible vorticity on a lifting body requires the presence of trailing vorticies which extend to infinity behind the wing.

A unique solution to the potential flow problem for a lifting wing when using distributed vorticity on the wing surface requires the following boundary conditions: one component of velocity at the surface (usually used to enforce tangential flow), and the circulation at each spanwise station of the wing (Kutta condition). The tangential condition at the wing surface is enforced by specifying that its local normal perturbation velocity cancel the normal component of free stream velocity. The Kutta condition requires a circulation distribution from the leading to trailing edges of the wing such that zero loading occurs at the latter. The use of a singularity induced perturbation velocity implicitly enforces the recovery condition of the freestream velocity infinitely far away since the singularity influence decreases with distance.

Green's third identity has reduced the problem from a p.d.e. in 3-dimensions to an equivalent distribution of vorticity on the boundary retaining the original boundary conditions. Panel methods use a finite element approach to this equivalent problem by discretizing the boundary (wing surface) into (usually) flat, finite, areas with a

predetermined vorticity distribution of unknown strength. Boundary conditions are applied at a number of discrete points resulting in a set of simultaneous equations which determine the strength of the vorticity distribution on each panel. In the limit of an infinite number of panels, each panel represents an area element of integration and the solution is exact.

CHAPTER 4

CALCULATION OF POTENTIAL FLOW

The calculation of a potential flow is central to this method. At each iteration this panel method calculates the potential velocity at each control point given the geometry of the wing, wake, and tunnel. Note that while the method determines the wake iteratively, at each iteration a wake shape must be assumed in order to calculate the potential flow. An analysis of the velocity distribution is then used to update the wake geometry. This iterative process continues until the solution converges. The convergence criteria is satisfied when the wake boundaries are aligned with streamlines and the flow on the wing upstream of the separation locus remains attached for the then present pressure distribution.

The steps in the formulation of the present panel method are as follows:

1) Determination of the panel geometry and singularity distribution.

2) Placement of the panels to represent the wing, wake and tunnel.

3) Evaluation of the induced velocities at each control point for unit values of each vorticity distribution.

4) Mathematical statement of the boundary constraints at the control points in the form of a matrix equation.

5) Solution of the matrix equation to obtain the panel strengths.

6) Calculation of the velocities at the control points that correspond to the panel strengths.

PANEL FORMULATION

The basic element of the method is the panel, which is defined by its shape and singularity distribution. An exact solution requires both the shape of and vorticity distribution on the boundary of the potential flow field to be completely correct. For example, modeling a 2-dimensional inclined flat plate using one panel with a uniform distribution of vorticity, duplicates the exact geometry of the flat plate, but the constant vorticity distribution limits the accuracy of the solution. Thus, the solution accuracy is limited by both the ability to represent the proper shape and the modeling of a singularity distribution with a finite number of panels.

Correspondingly, the complexity of a formula for the induced velocity of a panel varies with the complexity of both the panel geometry and the singularity distribution. This requires relatively simple panel geometry and singularity distributions. The panel planform (Figure 4-1) used here is a parallelogram with the equivalent of an infinite number of vortex filaments of infinitesimal strength distributed

parallel to the leading and trailing edges, and with the circulation density per unit length varying linearly in the chordwise direction. An unsuccessful attempt was made to calculate a closed form solution for the induced velocity of a panel that also was tapered. The difference in the computational effort required to evaluate a formula in closed verses open form results in preference being given to closed form solutions for the induced velocity of a panel. The parallelogram panel was chosen for its ability to model both the geometry and singularity distribution with a closed form induced velocity function. If a tapered panel is required, it is approximated by a swept panel of equal area that is swept by the same amount at the 1/2 panel chord position. If used alone, the panels would violate Helmholt'z vortex theorem, which states that vortex filaments cannot end in the fluid. The vortex filaments bend at the sides of the panel and form trailing vortices.

PANELING

Two considerations when determining paneling are the representation of the physical boundary location and the allowance for resolution of the singularity distribution. Wing panels are distributed evenly in the spanwise direction. A half cosine distribution concentrates panels at the leading

edge of the wing. The projection of the leading and trailing edges of the panels along the wing chord are given by

$$\xi_{le} = 1 - \cos\left(\frac{inc - 1}{nc}\,\frac{\pi}{2}\right) \qquad [4\text{-}1]$$

$$\xi_{te} = 1 - \cos\left(\frac{inc}{nc}\,\frac{\pi}{2}\right) \qquad [4\text{-}2]$$

where

xi$_{le}$ = fractional wing chord location of panel leading edge

xi$_{te}$ = fractional wing chord location of panel trailing edge

inc = chordwise row number

nc = number of chordwise panels

Wing section coordinates may be found in Reference 18. A closed form solution was used here for the coordinates of NACA 4 digit airfoils and a lookup table was used for the coordinates of a NACA 64A005 airfoil. Wake panels trail from the upper and lower surfaces of the wing and are initially aligned with the free stream. They are evenly spaced in both directions and have a constant sweep angle such that the wake is swept at the same angle as the trailing edge of the wing. The wing planform is determined and then the wing sections are determined at evenly spaced spanwise locations up to and including the wing tip. A right handed X,Y,Z coordinate system is used where X is in the downstream direction, Y is in

the spanwise direction, and Z is determined by the right hand rule (Figure 4-2). The wing sections are translated within each X-Z plane to accommodate sweep, taper and dihedral. The sections are then connected to form a collection of swept and possible tapered panels. The entire wing is then rotated about the Y axis at the inboard leading edge by the angle of attack. The wing can than be translated in the X and Z directions to bring it to the proper location in respect to the tunnel and possible other wings in a particular test.

An elliptic cross section tunnel with a possible ground board of arbitrary height has been modeled. The tunnel paneling is similar to that used by both Olechowski (14) and Tennison (15) with the addition of a ground board. A tunnel test in which a wing is horizontal has a right-left vertical symmetry plane as the center plane of both the wing and tunnel. In a test with a half wing placed vertically on a ground board, the ground board acts as a reflecting plane. The ground board is modeled implicitly by modeling an image half wing and tunnel. This effectively results in an equivalent full span wing in a modified tunnel shape (see Figure 4-3). The nomenclature for the tunnel is similar to that for a wing: there are nc panels along the length of the tunnel starting at the test section inlet and ns panels in the spanwise wing direction starting at the symmetry plane. The origin of the global X,Y,Z coordinate system is located at the

intersection of the X-Z symmetry plane, the test section inlet, and the X-Y symmetry plane of the tunnel. Figure 4-4 illustrates the tunnel nomenclature and relation of tunnel coordinates to global coordinates. The tunnel panel locations are figured in tunnel coordinates $X^{\sim},Y^{\sim},Z^{\sim}$ with the origin located at the center of the elliptic cross section at the entrance of the tunnel.

The tunnel cross section ($Y^{\sim}$-$Z^{\sim}$ plane) is invariant in the $X^{\sim}$ direction. The projection of the tunnel panel edges and control points in the $X^{\sim}$ direction on the plane $X^{\sim}=0$ are calculated for the quarter plane $Y >= 0$, $Z >= 0$ (global) and then reflected into the other quadrants. The control points are placed such that radii extended from the elipse center ($Y^{\sim}=0, Z^{\sim}=0$) to each of these control points are separated by equal angles. Then the panels are placed tangent to the surface at the control points such that their ends touch either each other or the reflection plane (Figure 4-3). The locations of the control points in the $Y^{\sim}$-$Z^{\sim}$ plane are given by

$$y_c = \frac{\text{sign}(\cos(\omega))\ Ht}{\text{sqrt}\left(\left(\frac{Ht}{B}\right)^2 + \tan^2(\omega)\right)} \qquad [4\text{-}3]$$

$$z_c = y_c \tan(\omega) \qquad [4\text{-}4]$$

where

$$\omega = \omega_0 \left(1 - \frac{2\,ins - 1}{2\,ns} \right) \qquad [4\text{-}5]$$

$$\omega_0 = \tan^{-1}\left(\frac{-B + SBH}{Ht\ \sqrt{1 - \left(\frac{-B + SBH}{B} \right)^2}} \right) \qquad [4\text{-}6]$$

and

SBH = distance from symmetry plane to tunnel elipse center

ins = spanwise tunnel panel number

ns = total number of spanwise tunnel panels

ω = angle of ray from the ellipse center to the control point

y_c , z_c = control point location in Y~-Z~ plane

Using the locations of the control points and the slope of each panel the edge points of the panels can be found. Symmetry indicates that the first panel edge is on the line Y~=-B+SBH and the last panel edge is on the line Z~=0.

$$sl = sign(-\tan(\omega)) \frac{Ht\ yc}{B\ \sqrt{B^2 - y_c^2}} \qquad [4\text{-}7]$$

$$yp = \frac{(z_c - sl\ y_c) - (z'_c - sl'\ y'_c)}{s' - s} \qquad [4\text{-}8]$$

$$zp = sl\ y_p + (z_c - sl\ y_c) \qquad [4\text{-}9]$$

where

sl = slope of panel

$()'$ = value of adjacent inward panel

y_p , z_p = panel edge location in $\bar{Y}$,$\bar{Z}$ frame

The tunnel cross sections were placed at equal intervals and panel edge points were connected to form the tunnel panels.

CALCULATION OF INDUCED VELOCITIES

The nondimensional circulation per unit length γ varies linearly along the panel chord and can be expressed as a function of ξ, where ξ is the fractional distance along the panel chord (Figure 4-1). γ_a and γ_b are the values of γ at the leading and trailing edges of the panel respectively.

$$\gamma(\xi) = \gamma_a (1 - \xi) + \gamma_b \xi \qquad [4\text{-}10]$$

Using the principal of superposition the vorticity distribution can be broken down into its basic elements. When refering to the influence of γ_a and γ_b, they represent the distributions due to the first and second terms of Equation 4-10 respectively (see Figure 4-5). Induced velocity functions will be calculated for unit values of the nondimensional parameters Γ, γ_a and γ_b (circulation and circulation per unit length respectively).

The panel illustrated in Figure 4-1 is useful to refer to for the velocity influence function derivation. By reversing the sign of the sweep angle Λ and rotating the panel 180 degrees about the line ($z'=0$ $x'=c'/2$), the leading and trailing edges are reversed. Thus, given the induced velocity function of either γ_a or γ_b, the induced velocity function of the other can be calculated from this rotation.

given

$$V = V (\lambda ,x',y',z')$$

then

$$\begin{aligned} u &= - u (- \lambda ,1-x',y',-z') \\ v &= v (- \lambda ,1-x',y',-z') \\ w &= - w (- \lambda ,1-x',y',-z') \end{aligned} \qquad [4\text{-}11]$$

where

$$\lambda = \tan (\Lambda)$$

The integrals for the induced velocity functions will be derived and presented here However, details of the integrations are collected in Appendix A. The panel coordinate system, denoted by a prime, is centered at the inboard side of the leading edge of a panel lying in the X'-Y' plane. The induced velocity function of a vortex filament is found using the double prime coordinate system shown in Figure 4-6. Using the Biot-Savart law the effect at point P of a finite vortex filament of unit strength aligned with the Y"

axis is

$$V = \frac{1}{4\pi} \int_{\eta=0}^{\eta=b''} \frac{\sin(\phi)}{r^2} \, d\eta \qquad [4\text{-}12]$$

where

$$r_p^2 = x''^2 + z''^2$$

$$r^2 = r_p^2 + (y'' - \eta)^2$$

$$\sin(\phi) = \frac{r_p}{r}$$

therefore

$$V = \frac{1}{4\pi} \int_{\eta=0}^{\eta=b''} \frac{r_p}{\left[r_p^2 + (y'' - \eta)^2 \right]^{(3/2)}} \, d\eta \qquad [4\text{-}13]$$

integrating (Reference 24 was used for integrals)

$$V = \frac{1}{4\pi r_p} \left[\frac{Y_1}{d_1} - \frac{Y_2}{d_2} \right] \qquad [4\text{-}14]$$

where

$$Y_1 = y'' - b''$$

$$Y_2 = y''$$

$$d_1^2 = Y_1^2 + r_p^2$$

$$d_2^2 = Y_2^2 + r_p^2$$

The components of velocity in the double prime frame are

$$u'' = \frac{z''}{4\pi r_p^2}\left[\frac{Y_1}{d_1} - \frac{Y_2}{d_2}\right] \qquad [4\text{-}15]$$

$$v'' = 0 \qquad [4\text{-}16]$$

$$w'' = \frac{-x''}{4\pi r_p^2}\left[\frac{Y_1}{d_1} - \frac{Y_2}{d_2}\right] \qquad [4\text{-}17]$$

Noting that the circulation distribution of the panel due to Γ is given by

$$\Gamma = \gamma\, d\xi = \gamma_b\, \xi\, d\xi \qquad [4\text{-}18]$$

then the induced velocity function of the panel shown in Figure 4-1 due to a unit value of the nondimensional parameter γ_b is

$$u'_b = \frac{1}{(1+\lambda)} \int_{\xi=0}^{\xi=1} u'' \qquad [4\text{-}19]$$

Redefining distances in panel coordinates which are denoted by a single prime and illustrated in Figure 4-1.

$$z'' = z'$$

$$x''^2 = \frac{(x' - \xi - \lambda y')^2}{1 + \lambda^2}$$

$$r'^2 = y'^2 + z'^2$$

$$d_1^2 = (x' - \xi)^2 + r'^2$$

$$d_2^2 = (x' - \xi - \lambda b)^2 + (y' - b)^2 + z'^2$$

$$Y_1^2 = (d_1^2 - r'^2_p)_{z'=0}$$

$$= \frac{(\lambda (x' - \xi) + y')^2}{1 + \lambda^2}$$

$$Y_2^2 = (d_2^2 - r'^2_p)_{z'=0}$$

$$= \frac{(\lambda (x' - \xi - \lambda b) + (y' - b))^2}{1 + \lambda^2}$$

then Equation 4-19 becomes

$$u' = \frac{z'}{4\pi} \int_{\xi=0}^{\xi=1} \frac{\xi \ \mathrm{sqrt}(1 + \lambda^2)}{(x' - \xi - \lambda y')^2 + z'^2(1 + \lambda^2)} \left[\frac{Y_1}{d_1} - \frac{Y_2}{d_2} \right] d\xi \qquad [4\text{-}20]$$

The two terms in the brackets of Equation 4-20 differ by a coordinate translation. Rewriting Equation 4-20 and using a translation results in

$$u'_b = u_1 - u_2$$

$$= u(\lambda_1, x', y', z') - u(\lambda_1, x' - \lambda b, y' - b, z') \qquad [4\text{-}21]$$

As shown in the appendix, u_1 integrates to

$$u_1 = \frac{1}{4\pi} \left(-(x' - \lambda y') F1 - z' \left(\lambda^2 + 1 \right)^{(1/2)} G1 + z' \lambda \, G2 \right) \qquad [4\text{-}22]$$

where

$$F1 = \left[\tan^{-1} \left(\frac{z' \left((x' - \xi)^2 + r'^2 \right)^{(1/2)}}{\lambda r'^2 - y'(x' - \xi)} \right) \right]_{\xi=0}^{\xi=1} \qquad [4\text{-}23]$$

$$G1 = \left[\sinh^{-1} \left(\frac{\lambda (x' - \xi) + y'}{\left((x' - \lambda y' - \xi)^2 + z'^2 (1 + \lambda^2) \right)^{(1/2)}} \right) \right]_{\xi=0}^{\xi=1} \qquad [4\text{-}24]$$

$$G2 = \left[\sinh^{-1} \left(\frac{(x' - \xi)}{r'} \right) \right]_{\xi=0}^{\xi=1} \qquad [4\text{-}25]$$

The Y′ component of velocity differs from the X′ by the constant factor $-\lambda$ due to the constant sweep angle of the vortex filaments.

$$v'_b = -\lambda\, u'_b \qquad [4\text{-}26]$$

Equation 4-21 illustrates explicitly that the two terms in the original integral for u′b , Equation 4-10, differ by a coordinate translation. This holds true for all three components of induced velocity.

$$V_b = V(\lambda, x', y', z') - V_1(\lambda, x' - \lambda b, y' - b, z') \qquad [4\text{-}27]$$

The Z′ component of induced velocity requires the funtion w

$$w_1 = \frac{z'}{4\pi} \int_{\xi=0}^{\xi=1} \frac{\xi\,(1+\lambda^2)^{(1/2)}\,(x' - \xi - \lambda y')}{(x' - \xi - \lambda y')^2 + z'^2(1+\lambda^2)} \left[\frac{Y_1}{d_1} - \frac{Y_2}{d_2}\right] d$$

[4-28]

integrating Equation 4-28

$$w_1 = \frac{1}{4\pi} \{ z'(1+\lambda^2)\, F1 - \lambda\, F2 - (x' - \lambda y')(1+\lambda^2)^{(1/2)}\, G1 + ((x' - \lambda y')\lambda - y')\, G2 \} \qquad [4\text{-}29]$$

where

$$F2 = \left[\left((x' - \xi)^2 + r'^2 \right)^{(1/2)} \right]_{\xi=2}^{\xi=1} \qquad [4\text{-}30]$$

The influence of a panel must include the effects of a complete vortex line filament, whereas only segments which are bound to the panel contribute to the induced velocity expressions that so far have been calculated above. Three pieces of the overall induced velocity for a complete set of the vortex filaments associated with γ_a and γ_b are still needed. As shown in Figure 4-7, they are 1) the filaments of quadratic strength along the sides of the panel from which they came, 2) the constant strength vortex elements that trail from their panel sides, and 3) the vortex filaments that extend downstream from the end of the wake to infinity (parallel to the X axis.)

First consider the quadratic strength elements. Along the panel sides the number of vortex filaments increase linearly and the strength of each vortex filament increases linearly also. This results in vortex filaments of quadratic strength at the sides of the panels and is the shed vorticity due to γ_b . The effect of the bound side trailing vortex filaments for γ_a is found by considering the γ_a vortex filaments to trail forward to the leading edge of the panel forming a similar quadratic strength vortex. The γ_a vortex

filaments then turn 180 degrees and trail in the X′ direction at constant strength. The circulation for any part of the panel is given by:

$$\Gamma = \gamma \, d\xi \qquad [4\text{-}31]$$

For the inboard side of the panel the vortex circulation is the sum of all contributions between x′= 0 and ξ. Thus, the strength of the bound side vortex for a unit γ_b is:

$$\Gamma(\xi) = \int_{\xi'=0}^{\xi'=\xi} \gamma \, d\xi' = \int_{\xi'=0}^{\xi'=\xi} \xi' \gamma_b \, d\xi'$$

$$= \frac{1}{2} \xi^2 \qquad [4\text{-}32]$$

Equation 4-14 is applicable with a coordinate rotation and changing the strength from uniform to a function of ξ. The following integral for a quadradic side vortex results:

$$V_{qs} = \frac{-r'}{4\pi} \int_{\xi=0}^{\xi=1} \frac{((1/2)\,\xi^2)}{(r'^2 + (x'-\xi)^2)^{(3/2)}} \, d\xi \qquad [4\text{-}33]$$

Integrating 4-33 gives:

$$V_{qs} = \frac{r'}{2}\left[\frac{x'}{r'^2}\left(\frac{(x'-\xi)^2}{r'^2}\right)_{\xi=0}^{\xi=1} + \left(\frac{x'+1}{r'^2\left(r'^2+(x'-\xi)^2\right)^{1/2}}\right)_{\xi=0}^{\xi=1}\right] + \frac{r'}{2}\,G2 \qquad [4\text{-}34]$$

The u′, v′, and w′ components are:

$$u' = 0 \qquad [4\text{-}35]$$

$$v' = -\frac{z'}{r'}\,V_{qs} \qquad [4\text{-}36]$$

$$w' = \frac{y'}{r'}\,V_{qs} \qquad [4\text{-}37]$$

The uniform strength vortex lines that lie along panel sides to represent vorticity trailing from upstream panels requires only a coordinate transform of equations 4-15 to 4-17 to imply the induced velocities.

$$u' = 0 \qquad [4\text{-}38]$$

$$v' = -\frac{z'}{2}\left[\frac{x'-\xi}{r'^2\left((x'-\xi)^2+r'^2\right)^{1/2}}\right]_{\xi=0}^{\xi=1} \qquad [4\text{-}39]$$

$$w' = \frac{y'}{2}\left[\frac{x'-\xi}{r'^2\left((x'-\xi)^2+r'^2\right)^{1/2}}\right]_{\xi=0}^{\xi=1} \qquad [4\text{-}40]$$

The semi-infinite trailing vorticity to infinity is assumed to be always parallel to the X axis (global coordinates). It is the only velocity function calculated in

global coordinates. This requires a coordinate transform of equations 4-15 to 4-17 and taking the limit as b" goes to infinity.

$$u = 0 \qquad [4\text{-}41]$$

$$v = \frac{-z}{y^2 + z^2}\left[-1 - \frac{x}{(y^2 + z^2)^{(1/2)}}\right] \qquad [4\text{-}42]$$

$$w = \frac{y}{y^2 + z^2}\left[-1 - \frac{x}{(y^2 + z^2)^{(1/2)}}\right] \qquad [4\text{-}43]$$

Note that this equation is nondimensional in units of the wing chord. The panel induced velocity equations use the panel chord as a reference length and the velocities are scaled to the wing chord before being combined.

MATRIX EQUATION

The induced velocity functions are used to calculate the induced velocity matrices. The velocities at each control point expressed in matrix notation are

$$\{u\} = [u]\{X\} \qquad [4\text{-}44]$$

$$\{v\} = [v]\{X\} \qquad [4\text{-}45]$$

$$\{w\} = [w]\{X\} \qquad [4\text{-}46]$$

$$\{u'\} = [u']\{X\} \qquad [4\text{-}47]$$

$$\{v'\} = [v']\{X\} \qquad [4\text{-}48]$$

$$\{w'\} = [w']\{X\} \qquad [4\text{-}49]$$

where

$\{u\}$ = Column vector of u velocities at all control points

$\{v\}$ = Column vector of v velocities at all control points

$\{w\}$ = Column vector of w velocities at all control points

$[u]$ = Matrix of u influence cooeffients

$[v]$ = Matrix of v influence cooeffients

$[w]$ = Matrix of w influence cooeffients

$\{X\}$ = Column vector of panel edge circulation / unit length

The prime indicates that each row is in the local panel coordinates of that rows control point.

The induced velocities due to the following pieces of singularity distribution at each panel are calculated in the local panel coordinates.

1) The bound panel vorticity

2) The bound side vorticity

3) The side vorticity trailing from upstream panels

These velocities are converted to global coordinates and added into the matices $[u]$, $[v]$, and $[w]$ at the row of the control point and column of the panel edge associated with the inducing vorticity distribution. Then the induced velocities

of all of the panels due to the semi-infinite vorticies trailing from the end of the wake are added to the matrices. The rotation matrix [R] is used to change the orientation of the induced velocities from the local frame to the global frame.

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = [R] \begin{Bmatrix} u' \\ v' \\ w' \end{Bmatrix} \qquad [4\text{-}50]$$

where

$$[R] = \begin{vmatrix} \cos\theta & -\sin\chi\sin\theta & -\cos\theta\sin\chi \\ 0 & \cos\chi & -\sin\chi \\ \sin\theta & \sin\chi\cos\theta & \cos\theta\cos\chi \end{vmatrix}$$

and

θ = incidence angle of panel from global coordinates

χ = dihedral angle of panel from global coordinates

The matrices [u'], [v'], and [w'] are calculated from [u], [v], and [w] by appling the inverse of the rotation matrix [R] at each control point.

$$\begin{Bmatrix} u' \\ v' \\ w' \end{Bmatrix} = [R]^{-1} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \qquad [4\text{-}51]$$

where

$$[R]^{-1} = \begin{vmatrix} \cos\theta & 0 & \sin\theta \\ -\sin\theta\sin\chi & \cos\chi & \cos\theta\sin\chi \\ -\sin\theta\cos\chi & -\sin\chi & \cos\theta\cos\chi \end{vmatrix}$$

θ = incidence angle of control point panel from global coordinates

χ = dihedral angle of control point panel from global coordinates

The condition of tangential velocity at the control points is expressed as

$$[w'] \{x\} = \{ -Vn \} \qquad [5\text{-}52]$$

where

-Vn = Minus the local normal component of the freestream velocity

The tangential flow condition is applied at all tunnel panel control points (Figure 4-4). At each spanwise station of the tunnel there are two more panel edges than there are panels. The strength of the vorticity per unit length is explicitly set to zero for the trailing edge of the last ring of tunnel panels. This Kutta like condition results in an equal number of unknowns and boundary conditions for the tunnel.

On a wing the boundary conditions depend upon flow attachment, and if not attached, the location of the separation point. For the separated case (Figure 4-2) the tangential flow condition applies to all control points located on wing panels in the attached flow region. At each spanwise location there is one more panel edge located on the wing in the attached flow region than control points. The

vorticity at the most downstream upper and lower panel edges on the wing are set equal and opposite.

$$\gamma_{uwe} + \gamma_{lwe} = 0 \qquad [4\text{-}53]$$

where

γ_{uwe} = distributed vorticity value at the trailing edge of last upper panel on wing.

γ_{lwe} = distributed vorticity value at the trailing edge of last lower panel on wing.

The vorticity of the trailing wake panels is set equal to the vorticity at the point where the wake leaves the wing. Thus, the separated wing has an equal number of unknowns and boundary conditions. For the ith wake panel

$$\gamma_{uwe} - \gamma_{uiwake} = 0 \qquad [4\text{-}54]$$

$$\gamma_{lwe} - \gamma_{liwake} = 0 \qquad [4\text{-}55]$$

where

γ_{uwe} = distributed vorticity value at the trailing edge of last upper panel on wing.

γ_{uiwake} = distributed vorticity value at the trailing edge of upper wake panel i.

γ_{lwe} = distributed vorticity value at the trailing edge of last lower panel on wing.

γ_{liwake} = distributed vorticity value at the trailing edge of lower wake panel i.

Originally in this work the treatment of the attached case was the same as the separated case with separation taking place at the trailing edge. The Kutta condition was enforced since the sum of the upper and lower vorticity at trailing edge is zero. However, large accelerations of the flow at the trailing edge was found to result for the cases tested with the number of spanwise stations larger than the aspect ratio. This resulted due to very large values of vorticity at the trailing edge. An examination of the solution matrix reveals that the influence of the wing trailing edge vortictiy on the normal velocity of the last upper and lower control points is small compared with the influence of upstream panels in the same spanwise strip. Thus, large values of vorticity are generated for the trailing edge nodes in order for them to have a comparable effect on the flow at the control points of the most downstream wing panels.

The trailing edge vorticity problem was resolved by setting the upper and lower trailing edge vorticity explicitly to zero. This results in one more equation than unknown if the tangential flow condition is enforced on all panels. The last upper and lower wing panels have the original tangential flow condition replaced with the condition of equal tangential velocity and thus, equal pressure. This condition is implemented by using the difference of the two rows of [u'] representing the local u' velocity at the upper and lower

trailing edge panel control points as a row of [A] and setting the difference to minus the difference of the u' component of free steam velocity at the two control points.

$$(\text{row } i\ [u'] - \text{row } j\ [u'])\ \{X\} = -(Vn_i - Vn_j) \qquad [4\text{-}56]$$

where

row i [u'] = the ith row of [u'] representing the u' velocity of the control point on last upper wake panel

row j [u'] = the jth row of [u'] representing the u' velocity of the control point on last lower wake panel

Vn_i = the local normal component of the free stream velocity at control point i

Vn_j = the local normal component of the free stream velocity at control point j

The vorticity on the wake panels is explicitly set to zero in the solution matrix. Thus, the solution matrix is the same size for either the attached or separated case. This results in an equal number of unknowns and equations for a wing with attached flow.

Since all of the allowable geometries possess symmetry about the Y=0 plane the number of of unknowns can be reduced by half. Then each unknown represents a vorticity distribution on both sides of the symmetry plane resulting in half of the original unknowns. Also, each boundary condition only need be applied to one of two symmetric points reducing

the number of equations by half.

MATRIX SOLUTION PROCEDURE

Gaussian elimination with partial pivoting and back substitution is used for the matrix solution (Reference 25). This method was implimented in a straight forward manner. Although a more efficient implimentation could speed this routine up by close to a factor of two, the execution time is a small fraction of the total and the improvement would be negligable for the present program.

Since the matrix solution method is of order (N^3) and the calculation of the induced velocities of order (N^2), for enough panels the matrix solution will dominate the computational effort needed for a solution and a faster (possible iterative) matrix solution procedure should be considered. At present the matrix solution time is small compared to the time needed to calculate the induced velocity matrices. The present method was implemented on a PDP 11-44 computer. Memory constaints limit the present implimentation to approximately 100 panels. Extrapolating the time needed for both calculating and solving the matrix indicates that the two calculation times would be equal at approximately 370 panels with the matrix in virtual memory, or at approximately

4000 panels if the computer used could address an entire matrix at the same time. The speedup in the matrix solution for directly addressable memory is based on the effect of implimenting the matrix solution procedure in main memory instead of virtual memory. This suggests that moving the program to a computer with more memory and address space would be the most efficient way to increase both the allowable number of panels and the execution time of the matrix solution procedure.

CHAPTER 5

BOUNDARY LAYER ANALYSIS

Potential flow methods neglect viscous effects and therefore are useless in regions where shear is significant. As recognized by Prandtl in 1904 the potential flow area and the shear layers can be considered as separate regions when the Reynolds number is large. Assuming a thin boundary layer which is of infinitesimal thickness leaves an inviscid flow about the body. The corresponding inviscid solution provides a velocity distribution at the edge of the boundary layer, and a boundary condition for the boundary layer equations. Some methods iterate this viscid/inviscid cycle, altering the shape of the body by the displacement thickness of the boundary layer until convergence is observed. The present method uses boundary layer analysis only to determine the separation locus on the wing.

The wing is divided into spanwise strips, within each of which the boundary layer is analyzed using 2-dimensional methods. The separation points then form the separation locus along the span of the wing. Along each strip the boundary layer is laminar from the stagnation point to the first occurence of either transition or separation. Laminar separation usually occurs near the leading edge of thin

airfoils and in the case of short bubble separation the bubble length is typically only 1 - 3% of the wing chord. Long bubble separation is possible in a real flow; however lacking a method to predict either the length or shape of a bubble, a laminar separation bubble of negligible length is assumed to form and then is followed by a turbulent boundary layer which reattaches to the wing. The boundary layer is modeled using Thwaites' (20) laminar analysis, and a modification of Michel's transition criteria as presented by Cebeci and Bradshaw (Reference 19). Then Stratford's (22) turbulent separation criterion is checked from the reattachment location to the trailing edge of the wing to determine if and where the boundary layer separates.

The use of the boundary layer analysis is described below and the details are described in the sections of this chapter which follow. The analysis is used to update the separation location. The iterative process consists of the following steps:

1) Initial configuration - attached flow with the wake trailing in the downstream direction.

2) Solve the inviscid flow and rotate the wake panels by $\tan^{-1}\left(\frac{Vn}{Vt}\right)$, repeating until $\frac{Vn}{Vt} < 0.01$.

3) Stop if the location of the separation point has converged. Otherwise, update the separation point and continue at step 2.

where

Vn = normal velocity at a wake panel control point

Vt = tangential velocity at a wake panel control point

The separation criteria was checked at each panel control point after carrying out code predictions for pressure distribution for a given configuration. When separation was indicated it was assumed to take place at the leading edge of the panel on which it was detected. Thus, on the next iteration such a panel and all downstream panels are treated as wake panels that adjust to the flow direction in order to lie on streamlines. This iterative process in conjunction with the calculation of the inviscid flow field detailed in Chapter 4 describes the complete modeling.

LAMINAR BOUNDARY LAYER

Thwaites' method (20) was chosen as one of the most accurate single parameter methods available. Although more accurate methods exist, they are significantly more complicated, and any method used would be limited by the accuracy with which the pressure distribution is known. A major limitation on the accuracy of the boundary layer analysis is in the relatively small number of points at which

the edge velocity is known.

The Navier-Stokes equations can be simplified for a thin boundary layer at large Reynolds numbers by eliminating terms that are small relative to the others. For the steady, incompressible, 2-dimensional boundary layer the equations reduce to:

continuity: $$u_x + v_y = 0 \qquad [5\text{-}1]$$

x-momentum: $$u u_x + v u_y = - p_x + (\nu) u_{y,y} \qquad [5\text{-}2]$$

$$= u_e u_{e,x} + (\nu) u_{y,y} \qquad [5\text{-}3]$$

y-momentum: $$p_y = 0 \qquad [5\text{-}4]$$

where the subscripts are defined as:

$()_x$ = partial derivative with respect to x

$()_y$ = partial derivative with respect to y

$()_e$ = edge value, between the viscous layer and the essentially potential flow region.

Equation 5-3 requires the additional assumption of a potential outer flow field. Here the x and y directions are intrinsic coordinates, i.e. parallel and perpendicular to the surface, respectively. Note that the effects of curvature have been neglected even though they may be significant in regions of high surface curvature. These are Prandtl's classical

boundary layer equations for 2-dimensional steady incompressible flow.

The Von Karman integral relation is obtained by integrating the x-momentum equation across the boundary layer and interchanging operations using Liebnitz' rule. This results in an ordinary differential equation in the x direction.

$$\frac{(\tau)_w}{(\rho)} = (u_e^2\,\theta)_x + \delta^* u_e u_{e,x} \qquad [5\text{-}5]$$

where

$$(\tau)_w = \text{shear stress at wall} = (\mu)\, u_{y,(y=0)} \qquad [5\text{-}6]$$

$$\delta^* = \text{displacement thickness} = \int_0^\infty \left(1 - \frac{u}{u_e}\right) dy \qquad [5\text{-}7]$$

$$\theta = \text{momentum thickness} = \int_0^\infty \frac{u}{u_e}\left(1 - \frac{u}{u_e}\right) dy \qquad [5\text{-}8]$$

For the following laminar discussion ue will indicate the edge velocity and a prime (') will denote the derivative with respect to x.

Thwaites noted that an extremely useful form can be obtained by multiplying Equation 5-5 by $\frac{u_e \theta}{\nu}$. A pressure gradient parameter is defined:

$$\lambda = \frac{\theta^2}{(\nu)} u' \qquad [5\text{-}9]$$

and a shape parameter H as:

$$H = \text{shape parameter} = \frac{\delta^*}{\theta} \qquad [5\text{-}10]$$

Then Equation 5-5 becomes:

$$\frac{(\tau_w)}{(\mu) u} \theta = \frac{u \theta}{(\nu)} + \frac{\theta^2}{(\nu)} u' (H + 2) \qquad [5\text{-}11]$$

Both sides of Equation 5-11 contain dimensionless boundary layer functions which are correlated by λ . Defining L as:

$$L(\lambda) = u \left[\frac{\lambda}{u'} \right]' \qquad [5\text{-}12]$$

and using the correlation based on both analytical solutions and experimental results

$$L(\lambda) = 0.45 - 6.0 \lambda \qquad [5\text{-}13]$$

Equation 5-11 has an explicit analytical solution when L is linear:

$$\theta^2 = \frac{0.45 (\nu)}{u^6} \int_0^x u^5 \, dx \qquad [5\text{-}14]$$

λ and H can be calculated from theta, which is found using the trapezoidal rule for integration. The displacement thickness and skin friction can then be determined from tabulated correlation functions. In practice the only parameters of interest are the momentum thickness and the value of λ when the skin friction vanishes indicating separation. The test for separation is Equation 5-15 below which is checked at each panel and indicates separation when the inequality is satisfied:

$$\lambda <= 0.2 \qquad [5\text{-}15]$$

If the boundary layer experiences transition before laminar separation occurs then the laminar analysis is terminated.

TRANSITION

Transition, the process of change from laminar to turbulent flow, occurs for sufficiently high Reynolds numbers. While lacking a fundamental theory for the location and length of the transition region, empirical methods do work reasonable well. The method chosen here is presented by Cebeci and Bradshaw (19) and is a modification of Michel's empirical relation using Smith and Gamberoni's correlation curve.

$$Re_{\theta,tr} <= 1.174 \left(1 + \frac{22{,}400}{Re_{x,tr}} \right) Re_{x,tr}^{0.46} \qquad [5\text{-}15]$$

When the inequality is satified the boundary layer is assumed to have experienced transition. Transition is assumed to take place over a negligible length.

TURBULENT BOUNDARY LAYER

Stratford's (22) method predicts separation of the turbulent boundary layer without solving the boundary layer equations. It provides vital information of where the separation point is located without requiring knowledge of (unknown) displacement thickness. Stratford divides the turbulent boundary layer into two separate regions, an outer layer in which the pressure is the driving force and an inner layer where inertial forces are balanced by shear forces.

In the outer layer the shear forces are small compared with either inertia or pressure forces. As a first approximation, the vorticity in the outer layer is neglected downstream of a point Xo and the flow is described by Bernoulli's equation.

$$\frac{\partial P}{\partial s} = 0 \qquad [5\text{-}16]$$

where

P = total pressure

s = distance along a streamline

therefore

$$P(X, \psi) = P(Xo, \psi) \qquad [5\text{-}17]$$

The dynamic pressure equals the dynamic pressure at Xo on the same streamline at Xo minus the rise in static pressure. Then the flow field in the outer region of the boundary layer can be found from the flow at Xo and the imposed pressure distribution.

Shear stresses decrease the total pressure on streamlines in a real flow. From the equations of motion

$$\frac{\partial P}{\partial s} = \frac{(\tau)}{y} \qquad [5\text{-}18]$$

and therefore

$$P(X, \psi) = P(Xo, \psi) + \int_{Xo}^{x} \frac{\partial(\tau)}{\partial y}\, ds \qquad [5\text{-}19]$$

Stratford considered a second flow in which the properties are the same at X=Xo, but which experiences a constant static pressure thereafter. The shear stress and velocity profiles are similar near X=Xo. The pressure gradient is assumed to

decrease the velocity at all points without affecting its slope in the outer region. Therefore, the total pressures of the two flows are nearly equal just downstream of Xo.

$$P(X, \psi) = P'(X, \psi) \quad (\psi) = \psi_{inner}) \qquad [5\text{-}20]$$

Where now a prime (') denote a value of the second flow and this equation applies only to the outer flow region.

Since

$$p' = \text{constant} = po \qquad [5\text{-}21]$$

where

p = static pressure

then

$$\frac{1}{2} \rho u^2(X, \psi) + p = \frac{1}{2} \rho u'^2 + po \qquad [5\text{-}22]$$

and

$$\frac{1}{2} \rho u^2(X, \psi) = \frac{1}{2} \rho u'^2 - (p - po) \qquad [5\text{-}23]$$

Thus, the dynamic pressure at any point is equal to the dynamic pressure in the comparison profile minus the static pressure rise in the neighborhood of Xo.

Stratford assumed that the outer layer has a flat plate turbulent boundary layer profile given by

$$\frac{u'}{u_p} = \left(\frac{y'}{\delta'} \right)^{(1/n)} \qquad [5\text{-}24]$$

$$\delta' = \frac{(n+1)\,(n+2)\,\theta'}{n} \qquad [5\text{-}25]$$

$$\theta' = 0.036 \times Re_x^{-(1/5)} \qquad [5\text{-}26]$$

Where n is slightly dependent upon Reynolds number. Differentiating Equation 5-24 with respect to Ψ results in a relationship that will be used to join the inner and outer layers.

$$\left[\frac{u}{y}\right]_{x,\,\Psi} = \left[\frac{u'}{y}\right]_{x,\,\Psi} \qquad (\Psi) = \Psi_{inner}) \qquad [5\text{-}27]$$

Within the inner layer the fluid has very little inertia and thus is greatly effected by the pressure gradient. At the body surface the inertia is zero, so pressure forces must be balanced solely by shear forces. Thus,

$$\frac{\partial p}{\partial x} = \frac{\partial(\tau)}{\partial y} \quad ; y = 0 \qquad [5\text{-}28]$$

which follows from the equations of motion. Assuming a pressure gradient starts at Xo, the velocity profile of the inner layer is distorted instantaneouslly, yet left unchanged at the wall where the no slip condition holds. There must be a smooth transition from the wall region (where pressure

forces and shear stresses are balanced) to the outer region (where pressure forces act to produce a decrease in dynamic head).

Using dimensional similarity arguments and requiring that the wall shear stress vanishes at separation, Stratford suggests that

$$u = \left(\frac{\nu}{\rho} \frac{\tau}{y} \right)^{1/3} F\left(\frac{1}{\rho} \left(\frac{\partial \tau}{\partial y} \right) \frac{y^3}{\mu^2} \right) \qquad [5\text{-}29]$$

where

$$y \left(\frac{\partial^2 \tau}{\partial y^2} \right) << \frac{\partial \tau}{\partial y}$$

Then with the assumption that relative motion is independent of viscosity in the fully turbulent part of the flow Equation 5-29 can be written as

$$u = A \left(\frac{y}{\rho} \frac{\partial \tau}{\partial y} \right)^{1/2} + B \left(\frac{\nu}{\rho} \frac{\partial \tau}{\partial y} \right)^{1/3} \qquad [5\text{-}30]$$

where A and B are constants.

For large values of $\frac{1}{\rho} \frac{d\tau}{dy} \frac{y^3}{\nu^2}$ Equation 5-30 is nearly

$$\frac{1}{2} \rho u^2 = \frac{1}{2} A^2 y \frac{d\tau}{dy} = \frac{1}{2} A^2 y \frac{dp}{dx} \qquad [5\text{-}31]$$

Thus very close to the wall an asymptotic form of the separating velocity profile exists where the velocity is proportional to square root of y and the dynamic pressure is proportional to y.

The constant A was determined by recourse to experiment. Its purpose is to act as a correction for the inaccuracy of the assumed velocity profile near the separation point. Stratford determined A to have the form,

$$A = \frac{2}{0.41\,\beta} \qquad [5\text{-}32]$$

where 0.41 is the flat plate value for the Karman constant and β is another constant. Thus, the inner layer is represented by the following profile,

$$\frac{1}{2}\,\rho\,u^2 = \frac{2}{(0.41\,\beta)^2}\,\frac{dP}{dx} \quad ; \quad (y < y_{inner}) \qquad [5\text{-}33]$$

With two velocity profiles, one valid in the outer layer (5-24) and the other valid in the inner layer (5-33), the final step is to find matching conditions at a suitable point.

Stratford chose the following matching conditions at the boundary between the inner and outer regions of the boundary layer:

$$y_{inner} = y_{outer} \qquad [5\text{-}34]$$

$$\psi_{inner} = \psi_{outer} \qquad [5\text{-}35]$$

$$\frac{d\,u_{inner}}{d\,y} = \frac{d\,u_{outer}}{d\,y} \qquad [5\text{-}36]$$

Then by defining the following equality at the boundary between the inner and outer layers

$$\psi_{outer}\left(\frac{d\,u_{outer}}{d\,y}\right)^3 = \psi_{inner}\left(\frac{d\,u_{inner}}{d\,y'}\right)^3 \qquad [5\text{-}37]$$

and noting that

$$\psi = \int_0^y u\,dy \qquad [5\text{-}38]$$

the following is obtained:

$$\left(\frac{y'}{\delta'}\right)^{\frac{2n-4}{n}} = \frac{3\,(0.41\,\beta)^4}{(n+1)\,(n\,\delta'\,\frac{dCp}{d\,x})^2} \qquad [5\text{-}39]$$

Furthermore, setting

$$\frac{u^2}{\psi_{outer}\,\frac{d\,u}{d\,y}} = \frac{u'^2}{\psi_{inner}\,\frac{d\,u'}{d\,y'}} \qquad [5\text{-}40]$$

results in

$$\frac{u^2}{u'^2} = \frac{3}{n+1} \qquad (\gamma_{outer} = \gamma_{inner}) \qquad [5\text{-}41]$$

Combining equations 5-23 and 5-24 and defining a non-standard pressure coefficient

$$Cp = \frac{P - P_p}{\frac{1}{2}\rho u_p^2} \qquad [5\text{-}42]$$

$$Cp = \left(\frac{y'}{\delta'}\right)^{2/n} \left(1 - \frac{u^2}{u'^2}\right) \qquad [5\text{-}43]$$

where

$$\gamma \rangle = \gamma_{inner}$$

and

$$Cp \langle = 1 - \frac{u^2}{u'^2}$$

Stratford's turbulent separation criterion combines Equations 5-24, 5-25, 5-34 to 5-36, and 5-41 into:

$$(2\,Cp)^{\frac{n-2}{4}} \left(x \frac{d\,Cp}{d\,x}\right)^{1/2} = 1.06\,\beta\,(10^{-6}\,Re_x)^{0.1} \qquad [5\text{-}44]$$

valid when $Cp <= \frac{n-2}{n+1}$.

The procedure is to march along the airfoil surface calculating both sides of Equation 5-44 at every x station. Separation is indicated when the equality is satisfied. The condition on the magnitude of Stratford's pressure coefficient comes about in order to properly match conditions at the interface between the inner and outer layers. Experimental results were used to determine the constants β and n to be 0.66 and 7 respectively. Stratford also made allowance for turbulent boundary layers with initial regions of favorable pressure gradient and also laminar boundary layers that transition to turbulence prior to separation.

An important parameter characterizing separation is the momentum thickness of the boundary layer at the position of minimum pressure (peak velocity). For a flow that is fully turbulent from the leading edge, Stratford proposed

$$Xo = \int_0^{Xo} \left(\frac{u}{u_p}\right)^3 dx \qquad [5\text{-}45]$$

where Xo is the distance from the equivalent leading edge. X is the actual distance from the leading edge and up is the

peak velocity.

For a boundary layer which is initially laminar, Stratford used Thwaites' expression for the laminar momentum thickness 5-14 combined with Equation 5-26 to determine the equivalent leading edge distance:

$$Xo = 38.2x_t \left(\frac{\nu}{x_t u_t}\right)^{(3/8)} \left(\frac{u_p}{u_t}\right)^{(1/8)} \left[\int_0^{x_t} \left(\frac{u}{u_p}\right)^5 d\left(\frac{x}{x_t}\right)\right]^{(5/8)} \quad [5\text{-}46]$$

This is based on the assumption of sudden transition. Note that the subscript "p" is used to specify the position or value of peak velocity, or the value at transition, whichever occurs first. The momentum thickness at either laminar separation or transition is used to determine the equivalent leading edge distance in the turbulent boundary layer routine.

CHAPTER 6

DISCUSSION OF RESULTS

A number of test cases are presented to illustrate and document the capabilities of the method. First, the ability of the method to accurately evaluate 2-D attached flow in free air is documented. The 2-D results actually were obtained for a wing of aspect ratio 1000. A large number of 3-D cases with attached flow were considered to determine the paneling constraints and the effects of wing geometry. Finally an analysis of 3-D wall interference and separation is presented.

2-D Attached Flow

Figure 6-1 illustrates the effect of the number of panels along the chord on the lift coefficient of a NACA 0012 wing section at an incidence of 6 degrees. The angle of attack is large enough to create a lift coefficient of order 1 and small enough that the flow over a real wing would stay attached over the entire surface of the airfoil (Reference 18). The theoretical value of the lift coefficient is 0.7175 (Reference 26). Figure 6-1 shows that the increase in the calculated lift coefficient is almost linear with the inverse number of chordwise panels for both the upper curve which is based on the circulation of the wing and the lower curve which is based

on a pressure summation. A linear extrapolation of the lift coefficients to values corresponding to an infinite number of chordwise panels implies results 2.0% and 1.3% greater than the theoretical value for the upper and lower curves respectively.

The upper curve in Figure 6-1 is based on applying the Kutta-Joukowski Theorem to a wing which is stated in Reference 21 as:

> An isolated two-dimensional airfoil in an incompressible inviscid flow feels a force per unit width of
>
> $$F = \rho_{inf} \; V_{inf} \; \text{cross} \; \Gamma \qquad [6\text{-}1]$$
>
> where Γ is the net circulation around the airfoil, and the direction of Γ is along the generator of the airfoil, the sense being determined by the right hand rule.

While the lift based on circulation is less dependent upon the number of chordwise panels used, it is only applicable for the case of a single 2-D wing. Attempting to applying Equation 6-1 to multiple bodies (including a wing and a tunnel) results in an answer that neglects the forces between the individual bodies due to the interaction of the bound vorticity; in the 3-D case the induced velocity of trailing vortices is neglected. For a vortex-lattice method with a single wing an analog of Equation 6-1 can be applied to the force on each vortex filament (Reference 21). An effective velocity induced by the rest of the vortex filaments replaces the free stream

velocity which includes the effects of the trailing vortices. Since Equation 6-1 is not applicable to multiple airfoils or 3-D wings the coefficient of lift will be based on the pressure summation for the rest of the results in this work.

Olechowski (14) and Tennison (15) used Equation 6-1 to calculate the lift of 3-D wings in a wind tunnel and presented results consistent with classical theory. Results were presented for a limited number of panels without attempting to extrapolate results for an infinite number of panels. A possible explanation for the agreement with classical theory is that the use of Equation 6-1 increased the lift coefficients due to the neglected downwash by an amount which roughly cancelled the error due to the finite paneling density. The force between the wing and tunnel due to the bound vortex interaction may have been relatively small compared to the lift of the wing due to the distance between the wing and the tunnel, and the relatively small values of of vortex density on the tunnel walls. Whether the proposed canceling of errors is random or systematic remains a topic of possible investigation.

The lower curve in Figure 6-1 is based on a summation of the pressure forces on each panel where the pressure for each panel is determined at the control point. The pressure coefficients were calculated in accordance with Bernoulli's

equation for a stead, incompressible irrotational flow where the velocities are nondimensionalized with respect to the freestream velocity.

$$C_p = 1 - u^2 - v^2 - w^2 \qquad [6\text{-}2]$$

The change in the minimum pressure coefficient is illustrated in Figure 6-2. The minimum pressure coefficient for a NACA 0012 wing section at an incidence of 6 degrees is plotted versus the inverse of the number of chordwise panels. The minimum pressure is particularly important for the boundary layer analysis which is limited by the accuracy within which the pressure distribution is known. The minimum pressure is reduced as the number of chordwise panels is increased resulting in the downstream adverse pressure gradient increasing, and thus, the boundary layer tending to separate closer to the leading edge.

Figure 6-3 shows a classic 2-D thin airfoil biplane as modeled in the present method. The thin airfoil section used is a NACA 0002 since the method requires airfoils to have thickness. The upper and lower wake panels are separated by a distance of the same order as the thickness of the trailing edges of the airfoils which is not distinguishable in the figure. The biplane is unstaggered with a separation of 1.0 chord between the airfoils. Figure 6-4 shows the effects of the paneling on Cl for both the upper and lower sections of

the biplane wing. The theoretical values of the lift coefficients for the two airfoils are 0.5936 and 0.5250 (Reference 27). The extrapolated values of the lift coeffients for an infinite number of panels are 1.35% greater and 0.50% lower than the theoretical values for the upper and lower airfoil sections respectively. This suggests that the method may apply to multiple wings with the limitation that each wing must kept clear of the wake and trailing vortices of the other wings. However, memory limitations of the PDP 11/44 on which the method was implemented constrain the number of panels that can be used such that paneling 3-D multiple wings in a tunnel is not practical.

3-D Attached Flow

Based on experience with the present method and the results in Reference 15 configuration 1 (Wing 1/Tunnel 1) is defined. Then the effects on CLfa (coefficient of lift in free air), CLt (coefficient of lift in tunnel) and the ratio CLt/CL due to variations of panel density and configuration geometry are examined.

Configuration 1

Wing 1

Aspect ratio	= 5
Angle of attack in degrees	= 6
Taper ratio	= 1.0
Sweep of wing in degrees	= 0.0
Dihedral angle in degrees	= 0.0

NACA wing section = 0012
Number of panels along chord = 4
Number of panels along semi-span = 2
Wake length in units of chord = 0.5
Number of wake panels = 2

Location of horizonal wing leading edge
4.5 chord lengths downstream of tunnel entrance
Centered in tunnel cross section

Tunnel 1

Tunnel length in units of wing chord = 10
Tunnel height / tunnel width = 1.0 (circular)
Tunnel width / wing span = 1.5
Number of panels / tunenl ring = 12
Number of panels along length = 4

Theoretical values of lift coefficient and ratio
CLfa = 0.5125 (Reference 26)
CLt = 0.5540 (Reference 2)
CLt/CLfa = 1.0810

Figure 6-5 demonstrates that the effect of the number of chordwise panels (nc) on a finite wing is similar to that for 2-D wings. The decrease in the lift at nc = 10 is due to the resolution of the pressure peak on the leading edge. For nc less than 10 the peak pressure occurs on the most upstream panel on the upper surface. At nc=10 the first panel is small enough to fit ahead of the pressure peak, thus the pressure difference between the upper and lower surfaces at the leading edge of the wing is reduced. The dependence of CL on the leading edge paneling should disappear as the number of panels becomes large enough to properly resolve the pressure peak on the leading edge. The pressure peaks on the leading edges for the 2-D cases shown in Figure 6-1 to 6-4 move back to the

second panel at nc = 4. Thus, there is no sudden change in the pressure distribution due to the panel spacing.

The ratio of CLt/CLfa shown in Figure 6-6 is unaffected by the shift of the pressure peak since it occurs for both CLt and CLfa. CLt/CLfa is approximately linear with the inverse of the number of chordwise wing panels.

Figure 6-7 shows the dependence of the lift on the number spanwise panels (ns) on the wing. The variation of lift with the number of spanwise wing panels is expected since a larger number of spanwise panels results in a better approximation of the spanwise loading of the wing being modeled. The lift ratio (CLt/CLfa) is relatively independent of the number of span panels as shown in Figure 6-8.

The ratio of the projected value of CLfa for an infinite number of chordwise panels and the value of CLfa for configuration 1 can be obtained from Figure 6-5 and an analogous ratio for spanwise paneling can be obtained from Figure 6-7. Then multiplying CLfa (configuration 1) by both ratios results in a projection of CLfa for an infinite number of both chordwise and spanwise panels. This procedure determines values of CLfa and CLt 22% lower than the theoretical value. The anomaly in the CL curves at nc = 10 due to the resolution of the peak negative pressure near the

leading edge suggests that values of nc significantly greater than 10 (possible 30 or larger) may be needed in order to project accurate values for an infinite number of chordwise panels for CLfa and CLt. However, the ratio of the extrapolated values, CLt/CLfa, is 0.83% higher than the theoretical value. This implies that while a large number of panels may be needed for absolute values of CLfa and CLt, the ratio may be obtained accurately with a relatively small number of panels.

The geometry of Configuration 1 was chosen such that the variation of CLt with the number of tunnel panels would be small compared to the variation with the number of wing panels in order to examine the effect of the wing paneling independent of the tunnel paneling. In order to examine the effects of the tunnel paneling which Configuration 1 was designed to reduce, Configuration 2 is defined. Configuration 2 differs from Configuration 1 ONLY in the ratio of tunnel span to wing span, therefore CLfa is the same value for both configurations. Reducing the span ratio (B/b) from 1.5 (Configuration 1) to 1.25 (Configuration 2) increases the interaction between the wing and tunnel, resulting in an increased sensitivity to the tunnel paneling and wing geometry.

Change from configuration 1 to configuration 2

Tunnel span / wing span = 1.50 (configuration 1)

Tunnel span / wing span = 1.25 (configuration 2)

Theoretical values of lift coefficient and ratio for configuration 2

CLfa	= 0.5125	(Reference 26)
CLt	= 0.5774	(Reference 2)
CLt/CLfa	= 1.1266	

Figure 6-9 shows the effect of the number of panels along the tunnel length (nct). The tunnel length is keep constant and the size of the panels is varied. The tangential flow boundary condition is only enforced at the panel control points such that the flow can leak though the panels except at the control points. Increasing the number of panels reduces the leakage through the tunnel walls and increases the effects of the tunnel. For configuration 1 doubling the number of lengthwise panels from 4 to 8 increases the ratio CLt/CLfa by a negligible 0.04%. Configuration 2 shows an increased sensitivity to the tunnel paneling.

Figure 6-10 shows the effect of the number of panels in each ring along the tunnel length. The ratio CLt/CLfa is almost independent of the number of circumferential panels for the circular tunnel with nst >= 2 (8 panels per ring). For an elliptic tunnel or a tunnel with a ground board the effect of the circumferential paneling is expected to be more pronounced since the shape the modeled tunnel cross section is more dependent upon the paneling.

Figure 6-11 shows the effect of the length of the modeled test section. Note that the panel size was kept constant and the number of panels along the length of the tunnel was altered. The tunnel interference is greatly reduced for a short tunnel (5 wing chord lengths), however the effect of lengthening the tunnel from 5 to 10 wing chords is less than 0.5% for both configurations (1 & 2).

For attached flow the influences of the wake length and paneling (Figures 6-12 to 6-15) are small compared to the effects of the wing and tunnel paneling. In Figures 6-12 and 6-13 the wake length is held constant; the size of the wake panels is held constant in Figures 6-14 and 6-15. Figures 6-12 and 6-13 show that the number of wake panels does not affect the lift for attached flow at a moderate angle of attack. For the attached case the wake panels allow the trailing vortices to follow the wake streamlines up to the end of the wake. The wake shape is a very gradual curve at small angles of attack and is adequately modeled by a small number of panels. It will be shown later in this chapter that for separated flow the choice of wake length and paneling has a much larger effect.

Figures 6-14 and 6-15 show the effects of the wake length on CLt, CLfa and CLt/CLfa. The longer the wake is the greater distance that the downstream end of the wake is deflected in

the -z direction. This places the trailing vortices closer to the tunnel and produces greater interference.

Figures 6-16 and 6-17 show the effects of the dihedral angle of the wing. The wing dihedral alters the location of the wing trailing vortices in relation to each other and the wind tunnel. The effect of dihedral is minimal in the circular tunnel, however for an elliptic test section the effect of the wing tips approaching the tunnel walls would increase the interference relative to a circular tunnel.

Figures 6-18 and 6-19 show the effects of the sweep of the wing. As the sweep increases the wing loading decreases and thus, the effect of the tunnel is reduced.

Figures 6-20 and 6-21 show the effects of taper. A decrease in the taper ratio causes a decrease in the wing loading at the wing tips for a wing of fixed aspect ratio and span. The wall interference is reduced as the wing tip loading is decreased.

Figures 6-22 and 6-23 demonstrates the effect of aspect ratio on CLt, CLfa, and CLt/CLfa. For a wing of fixed span the area of the wing and the wall interference decrease as the aspect ratio is increased.

Figure 6-24 shows the effect of the span ratio (B/b) on the interference correction. As the wing tips approach the walls the interference increases as expected. An important effect of decreasing the span ratio is the increasing number of panels needed to accurately model the walls. Bowcutt (Reference 16) used an image method to model the tunnel walls due the large number of tunnel panels needed to control leakage when the tunnel was explicitly represented by panels for extreme test conditions.

Figure 6-25 demonstrates the effect of the tunnel eccentricity for a constant span ratio (B/b). As the tunnel walls approach the wing the wall interference increases. However, for powering a wind tunnel the relevant parameters include the test section area rather than the test section span. Figure 6-26 shows the effect of the tunnel cross section eccentricity for a constant test section area. The values of eccentricity chosen (0, 0.681, 0.866) correspond to height/width ratios of 1, 0.75 (the value for the 7 1/2 by 10 foot wind tunnel at the Wright Brothers Facility at M.I.T.) and 0.5.

Figure 6-27 shows the effect of the height of a ground board on the ratio CLt/CLfa. The modeling of a ground board height of 0.5 tunnel height in a circular tunnel is equivalent to a tunnel without a ground board. Thus, Figure 6-27 shows a

variation of the ground board height for configuration 1. The lower curve shows the unexpected result of a value of CLt/CLfa less than 1.0. This result is indicative of the need to determine the effect of panel density on a particular geometry to accurately predict the wall effects. A wing with twice the number of chordwise panels (8 instead of 4) shifts the curve upward resulting in values of CLt/CLfa greater than 1.0. A projection of the value corresponding to an infinite number of panels would require many runs to determine the effects of paneling density. The effective increase in the ratio of tunnel area to wing area is expected to reduce the interference for a value of ground board height/tunnel height of 0.25. For a ground board height ratio of 0.05 the tunnel walls are very close to the root of the wing and increase wall interference. This suggests the use of the method to optimize the ground board height for a particular test.

Figures 6-28 to 6-30 illustrate the effects of wing geometry on the spanwise lift and circulation distributions. The values corresponding to the modeled wings are plotted showing the constant value of circulation at each spanwise station. For configuration 3 the spanwise paneling density is increased to resolve the spanwise variations and the aspect ratio is altered to allow comparison with the theoretical results in Reference 26.

Changes from configuration 1 to configuration 3

Tunnel span / wing span	= infinite
Aspect ratio	= 6
Panels along semi-span	= 5
Panels along wing chord	= 6
Wake panels	= 1

Figure 6-28 shows the dependence of the spanwise lift distribution on aspect ratio. Comparing the values of the local lift coefficients with theory at the span locations corresponding to the control points of the wing shows a difference of less than 1%.

Figure 6-29 shows the effect of sweep on the spanwise circulation distribution. The largest difference between the calculated and theoretical values is 28% and occurs at the most outboard control point of the wing with a sweep angle of 45 degrees. The reduction of the total circulation of the wing with sweep angle is not shown in Figure 6-29 due to the normalization by the value of circulation at the centerline of the wing.

The effect of the taper ratio on the spanwise circulation distribution is shown in Figure 6-30. Configuration 3 was used with the number of spanwise panels altered from 5 to 3 because the wake shape failed to converge for a greater number of spanwise panels. The unconverged wake shape remained almost aligned with the flow at each iteration as it moved up

and down slightly. The variation in the lift coefficient with iterations was less than 0.01%. Adding a damping factor in the wake rotation at each iteration may solve this problem, however time limitations prevented testing this proposed solution. Comparing the results of the untapered wing in Figure 6-30 (ns=3) with the corresponding wing in Figure 6-29 (ns=5) shows that the reduced number of spanwise stations increases the error at the most outboard control point from 20% to 27%. The error for the tapered wing in Figure 6-30 is 37%. However the theoretical value of circulation is smaller for the tapered panel resulting in a greater relative error when the absolute errors are both 20% of the reference value. This suggests that the taper ratio does not have a strong effect on the models ability to assume a proper spanwise distribution of circulation for a particular number of spanwise panels.

Separation

A number of cases were presented in Figures 6-13 and 6-14 to demonstrate the effect of wake length and paneling for the attached case. Figure 6-31 shows the effects of wake length and paneling density on the calculated lift coefficient for the separated case. A number of 2-D runs were made at a Reynolds number of 1.0 X 10E6 using a NACA 0012 airfoil with 10 panels along the chord at 17 degrees angle of attack. The

wake length of 1.0 wing chord was chosen as the longest wake that could practically be modeled with a small number of panels and still have the wake panels similar in length to the wing panels near the trailing edge. The variations in wake panel density resulted in a change in lift of 2% and 3% for the wakes of 1.0 and 0.5 wing chord lengths respectively. This suggests using a wake length of 0.5 chord lengths with 8 panels per chord length (4 panels total) to model separated cases. This paneling density was least affected by the wake length and the shorter wake requires fewer panels. It may be advantageous to use even fewer wake panels in cases where the number of panels in other locations has a greater effect on the calculation and it is necessary to reduce the total number of panels to fit within the limits of the computer code.

Configuration 4 was chosen to test the method's ability to model separation accurately. The configuration was chosen to correspond to experimential lift and pressure data presented in References 18 and 26 respectively. The 2-D free air case has the advantage of greater panel density compared to a 3-D case with a tunnel for a limited number of panels. The number of wake panels is set to 6 to resolve the wake shape. Figures 6-32 to 6-38 illustrate the effects of the angle of attack on the pressure distribution, separation point, lift coefficient, and wake shape.

Changes from configuration 1 to configuration 4

Reynolds number	= 1.2 x 10^6
Tunnel span / wing span	= infinite
Aspect ratio (2-D approximation)	= 1000
NACA wing section	= 2412
Number of panels along chord	= 20
Number of panels along semi-span	= 1
Number of wake panels	= 6

Figure 6-32 illustrates the pressure distribution of the configuration 4 wing at an angle of attack of -1.7 degrees. The flow is attached and the Kutta condition of equal pressure evident from the pressure distributions of the upper and lower surfaces meeting at the trailing edge control point. The relatively large wing panels near the trailing edge result in the pressure rise associated with the stagnation point at the finite angle trailing edge of an airfoil in an inviscid flow field extending over the most downstream one or two wing control points. The boundary layer analysis routine should probably predict separation due to the pressure rise for all airfoils if the analysis was continued to a stagnation point at the trailing edge. To allow the prediction of fully attached flow the boundary layer analysis was stopped one panel short of the trailing edge. Thus, it would not be possible to detect separation after the 85% chord point for the configuration 4 wing since that would have required a separtion prediction for the most downstream wing panel only.

Figure 6-33 shows the pressure distribution for the

configuration 4 wing at an incidence of 2.8 degrees. Experimental data from Reference 26 is shown for comparison. The experimental pressure coefficient data was read with an error of approximately 0.05 from a graph. Note that both the experimental and calculated pressure distributions cross such that the net pressure force near the trailing edge is directed downward. The calculated flow separates at the second to last panel (85% chord). Increasing the paneling density to 30 chordwise panels results in the separation point moving upstream to the 79% chord point 4 panels upstream of the trailing edge. The separation point may move upstream with the greater resolution of the negative pressure peak and increased adverse pressure gradient from that point to the separation point.

The algorithm of excluding the most downstream control point from the boundary layer analysis to prevent predicting separtion due to the stagnation point at the trailing edge may need modification. A possible modification would be to exclude a certain percentage of the airfoil ahead of the trailing edge (approximately 7%) or the two most downstream wing panels, whichever is greater. Once separation occurs the pressure distribution is modified over the entire airfoil. The adverse pressure gradient is generally increased as the separation point moves upstream since the distance along the airfoil between the pressure peak and the separation point is

decreased and the negative peak pressure coefficient and the pressure level of the separated region are less affected. This tends to cause the separation point to move upstream on the airfoil. Thus, inadvertently predicting separation at the trailing edge may cause separation over a significant portion of the airfoil to be predicted.

Figure 6-34 illustrates the pressure distribution for the configuration 4 wing at an angle of attack of 7.4 degrees. The flow separates at 47% of the chord. Increasing the paneling density to 30 chordwise panels results in the separation point moving upstream to the 37% chord point. The increased resolution of the negative pressure peak may be responsible for the upstream movement of the separation point with the increased panel density. The increased value of the negative pressure peak should reach a finite limit for an infinite number of panels based on Figure 6-2. Thus, as the number of panels approaches infinity, the separation point should move upstream on the airfoil a finite distance.

Both Figure 6-34 and 6-35 show the characteristic flat pressure distribution of the separated region. The separation point is farther upsteam in Figure 6-35 then in Figure 6-34, which is expected since the angle of attack is greater. However, Reference 26 indicates attached flow for both angles of attack (7.4 and 13.8 degrees). While the movement of the

location of the separation point with angle of attack and paneling density is following an expected trend, the predicted location does not agree with the experimental data.

The pressure level of the separated region has a smaller value when the separation point is farther upstream. As a typical case with separation iterates toward a solution the pressure distribution associated with a particular separation point will typically cause the boundary layer analysis to move the separation point upsteam by one panel. The pressure of the separated region decreases as the separation point moves upstream on the wing. In the classical solution of a 2-D thin airfoil the circulation decreased monotonically from the leading edge to the trailing edge. For this model the value of distributed circulation of the shear layer is determined to be equal to the value of the distributed circulation on the wing at the separation point. As the separation point moves upstream the value of the distributed circulation of the wake panels increases, causing the pressure in the separated region to decrease.

Figure 6-36 shows the pressure distribution of the configuration 4 wing at 17.9 degrees angle of attack and experimental values from Reference 26. This method predicts separation at 84% chord. This is inconsistent with the previous predictions of separation much farther upstream for

smaller angles of attack and disagrees with both References 18 and 26. Changing the number of chordwise panels from 20 to 30 results in the separation point moving upsteam to the 1.2% chord point (Figure 6-37). This is consistent with the previous separation predictions for angles of attack of 7.4 and 13.8 degrees. The anomaly of the separation prediction at 84% chord in Figure 6-36 may be due to the panel locations for this particular case. However, without an understanding of the mechanism, it is impossible to predict with certainty if the problem can be overcome by either a different distribution of panels or by using a larger number of panels.

Comparing the calculated and experimental pressure distributions in Figure 6-37 shows that the calculated separation point is upstream of the experimental separation point and that the calculated pressure of the separated region is higher than the experimental value. The calculated negative pressure peak is only -1.8 in Figure 6-37 compared to -11.9 in Figure 6-36. The agreement of the calculated and theoretical pressure distributions is closer for the lower surface of the airfoil than for the upper surface since the effect of the separation point is greater for the upper surface.

Figure 6-38 shows the initial wake shape and the converged and wake shape corresponding to the pressure

distribution in Figure 6-35 of the configuration 4 wing at an incidence of 13.8 degrees. The net leakage of the wing and wake is equal to the net flux through the gap at the end of the wake. For an infinite panel density there would be no net flux through the gap at the downstream end of the wake. The wake shape is not constrained and may be an improvement over the straight and parallel upper and lower wakes of Reference 15.

Figure 6-39 shows the convergence history corresponding to Figures 6-35 and 6-38. Arrows indicate iterations in which the wake shape is aligned with the flow. The wake shape is dependent on the separation point only for a particular configuration. The only effect of the Reynolds number is in the boundary layer analysis which determines the separation point. The separation point in this case moves only one panel at a time, except for between the second and third iteration when the flow changes from attached to separated two panels upstream of the trailing edge. At iteration 28 the solution is converged, because the separation point no longer moves upstream. This is responsible for the large change in the lift coefficient between iterations 2 and 3. The effect of the separation location is indicated by the iterations marked by arrows indicating the wake is aligned with the flow. In cases with separation the number of iterations required for convergence is approximately 4 times the number of panels

between the final separation location and the trailing edge.

In an attempt to demonstrate the effect of tunnel interference on the separation locus of a wing, configuration 5 was defined such that the presence of the tunnel would have a large effect on the wing and affect the separation locus of the wing. Configuration 5 corresponds to a case illustrated in Reference 15 in which separation locus was shown to move downstream near the tips of the wing in the tunnel compared to free air.

Changes from configuration 1 to configuration 5

Wing 5

Angle of attack in degrees	= 10
Number of panels along chord	= 6
Number of panels along semi-span	= 4
Number of wake panels	= 1

Location of horizonal wing leading edge
2.5 chord lengths downstream of tunnel entrance
Centered in tunnel cross section

Tunnel 5

Tunnel length in units of wing chord	= 5
Tunnel width / wing span	= 1.25
Number of panels / tunenl ring	= 8
Number of panels along length	= 2

Figures 6-40 and 6-41 illustrate the calculated pressure distributions of the configuration 5 wing in the tunnel and in free air. In both cases the flow is fully attached. The limitation of not being able to separate at the last panel would require separation to start at 40% of the chord or

sooner. The separation shown in Reference 15 occurred at the 70% and 85% chord positions. The cosine spacing of Reference 15 results in a high density of panels at the trailing edge as well as the leading edge allowing separation predictions close to the trailing edge. For this method the presence of the tunnel increases the negative pressure coefficient peaks by 0.25 for the most outboard station and by 0.28 for the rest of the spanwise stations, which correspond to 17% and 19% respectively. The most outboard station has a negative pressure coefficient peak 32% and 37% lower than the the most inboard station for the case with and without the tunnel respectively. The change in the pressure distributions for the wing in and out of the tunnel should affect the separation location, given that there is separation and that the number of chordwise panels is increased to a number that would allow the resolution of the separation location (estimate 30).

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

The present method provides a useful tool for the determination of wind tunnel wall effects for attached flow with general geometries which can only be approximated using classical methods. However, this method requires a number of cases be analyzed to determine and correct for the effects of the finite paneling density. The effort that this method requires is dependent upon the complexity of the geometry. Thus, this method could not be used for "real time" correction of wind tunnel data since a detailed study of a particular case is needed to generate corrections.

The performance of this method for wings experiencing extensive regions of separated flow is questionable and not as promising as the previous work of Tennison (15) in the ability to predict the separation locus. The significant differences between this and Tennison's method that may cause the discrepancy in the prediction of the separation locus are the chordwise spacing of the panels and the laminar boundary layer analysis method. The half cosine spacing was chosen for this work to prevent the chordwise dimension of the adjacent wing and wake panels at the trailing edge of the airfoil from differing in size by an order of magnitude or more. An abrupt

change in panel size did not occur in Tennison's work since the maximum chordwise paneling used by Tennison resulted in the smallest paneling being 5% of the chord length. Changes that might solve the separtion problem are switching to a different method for the laminar boundary layer analysis and/or repaneling the wing and wake in the chordwise direction such that there are concentrations of panels at the leading and trailing edges with a gradual variation of panel chord lengths.

For use of the present method to calculate corrections for separated flow, a possible additional test case would be to calculate the particular case in the tunnel with the separation locus determined from the wind tunnel test. Then the effect of the tunnel could be examined directly by calculating the velocities induced by the tunnel walls at the wing. The strength of the velocity induced by the tunnel walls could then be used to estimate the validity of the separation locus in free air.

For cases with close to the maximum number of panels the computer code for this method requires almost all of the available memory of the PDP 11/44 and any attempt to use the computer simultaneously results in the computer crashing. The largest cases take approximately 8 hours of computer time. If possible the computer code should be moved to a more suitable

computing environment. Running the computer code on either a VAX computer or a personal computer such as the Macintosh or IBM System 2 would increase the maximum number of panels possible and reduce the execution time of the program.

APPENDIX A

INTEGRATION OF SELECTED VELOCITY FUNCTIONS

The integrals that were set up in Chapter 4 in order to derive the induced velocity functions are integrated in this appendix. The integration of a function ul is needed to complete the statement of the panel u' velocity function in closed form.

$$u_1 = \frac{z'}{4\pi} \int_{\xi=0}^{\xi=1} \frac{\xi \ \mathrm{sqrt}(1 + \lambda^2)}{(x' - \xi - \lambda y')^2 + z'^2(1 + \lambda^2)} \frac{Y_1}{d_1} d\xi \qquad \text{[A-1]}$$

where

$$r'^2 = y'^2 + z'^2 \qquad \text{[A-2]}$$

$$d_1^2 = (x' - \xi)^2 + r'^2 \qquad \text{[A-3]}$$

$$Y_1^2 = \frac{(\lambda (x' - \xi) + y')^2}{1 + \lambda^2} \qquad \text{[A-4]}$$

In order to integrate u_1 key subexpressions are identified, Equation A-1 is expanded in terms of the key subexpressions, and the expanded terms are integrated. The integration varibles a, e, f, and g are used. First, the variable transformation (Equations A-5 and A-6) is performed.

$$a = x' - \xi - \lambda y' \qquad [A\text{-}5]$$

$$da = - d\xi \qquad [A\text{-}6]$$

$$a_1 = x' - \lambda y' \qquad [A\text{-}7]$$

$$a_2 = x' - \lambda y' - 1 \qquad [A\text{-}8]$$

$$e = \lambda (x' - \xi) + y' \qquad [A\text{-}9]$$

$$f = \left((x' - \xi - \lambda y')^2 + z'^2 (1 + \lambda^2) \right)^{(1/2)} \qquad [A\text{-}10]$$

$$g = \left((x' - \xi)^2 + r'^2 \right)^{(1/2)} \qquad [A\text{-}11]$$

Manipulating Equation A-1

$$u_1 = \frac{z'}{4\pi} \int_{a_1}^{a_2} \frac{\lambda e a}{f^2 \left((e - y')^2 + r'^2 \lambda^2 \right)} da$$

$$-(x' - \lambda y') \frac{z'}{4\pi} \int_{a_1}^{a_2} \frac{e}{a^2 + g^2} da \qquad [A\text{-}12]$$

and

$$u_1 = \frac{z'}{4\pi} \int_{a_1}^{a_2} \frac{1}{(1 + \lambda^2)^{(1/2)}} \cdot \frac{- \lambda a^2 + e a}{f^2 (f^2 + e^2)^{(1/2)}} da$$

$$+ \frac{z'}{4\pi} \int_{a_1}^{a_2} \frac{\lambda}{g}\, da$$

$$- \frac{z'}{4\pi} \int_{a_1}^{a_2} \frac{(a + y')\, q + g\, y'}{(z'^2 g^2 + q^2)\, g}\, da \qquad \text{[A-13]}$$

where

$$q = \lambda r'^2 - y' (x' - \xi) \qquad \text{[A-14]}$$

noting that

$$d e = e' = \lambda\, da \qquad \text{[A-15]}$$

$$d f = f' = -f\, a\, da \qquad \text{[A-16]}$$

$$d g = g' = g^{-1} (a + \lambda y')\, da \qquad \text{[A-17]}$$

$$d q = q' = - y'\, da \qquad \text{[A-18]}$$

Substituting the key expresions into Equation A-13 and integrating the second term results in

$$u_1 = \frac{z'}{4\pi} \frac{-1}{(1 + \lambda^2)^{(1/2)}} \int_{a_1}^{a_2} \frac{f^2 e' - e f f'}{f^2 (f^2 + e^2)^{(1/2)}}$$

$$+ \frac{z'}{4\pi}\frac{\lambda}{} \left[\sinh^{-1} \frac{a + \lambda y'}{r'} \right]_{a_1}^{a_2}$$

$$- \frac{(x' - \lambda y')z'}{4\pi} \int_{a_1}^{a_2} \frac{q\, g\, g' - g^2 q'}{(z'^2 g^2 + q^2)\, z'\, g}\, da \qquad \text{[A-19]}$$

Integrating the first term and manipulating the third term.

$$u = \frac{z'}{4\pi} \frac{-1}{(1 + \lambda^2)^{(1/2)}} \left[\sinh^{-1} \frac{e}{f} \right]_{a_1}^{a_2}$$

$$+ \frac{z'}{4\pi}\frac{\lambda}{} \left[\sinh^{-1} \frac{a + \lambda y'}{r'} \right]_{a_1}^{a_2}$$

$$- \frac{(x' - \lambda y')}{4\pi} \int_{a_1}^{a_2} \frac{q\, g\, z'\, g'\, z' - g\, z'^2 q'}{(z'^2 g^2 + q^2)\, z'\, g} \qquad \text{[A-20]}$$

Integrating the last term results in

$$- \frac{(x' - \lambda y')}{4\pi} \left[\tan^{-1} \frac{z'\, g}{q} \right]_{a_1}^{a_2} \qquad \text{[A-21]}$$

In terms of the original variables

$$u_1 = \frac{1}{4\pi}\left(-(x' - \lambda y')F1 - z'\left(\lambda^2 + 1\right)^{(1/2)} G1 + z' \lambda\, G2\right) \quad \text{[A-22]}$$

where

$$F1 = \left[\tan^{-1}\left(\frac{z' \left((x' - \xi)^2 + r'^2\right)^{(1/2)}}{\lambda r'^2 - y'(x' - \xi)} \right) \right]_{\xi=0}^{\xi=1} \quad \text{[A-23]}$$

$$G1 = \left[\sinh^{-1}\left(\frac{\lambda (x' - \xi) + y'}{\left((x' - \lambda y' - \xi)^2 + z'^2 (1 + \lambda^2)\right)^{(1/2)}} \right) \right]_{\xi=0}^{\xi=1} \quad \text{[A-24]}$$

$$G2 = \left[\sinh^{-1}\left(\frac{(x' - \xi)}{r'} \right) \right]_{\xi=0}^{\xi=1} \quad \text{[A-25]}$$

The integral for the third component of velocity differs from the integral for the first component by a factor of the integration variable a. The manipulation and integration of Equation A-22 is similar to that for Equation A-1.

$$w_1 = \frac{z'}{4\pi} \int_{a_2}^{a_1} \frac{a\,(a - (x' - \lambda y'))}{f} \frac{e}{g}\, da \quad \text{[A-26]}$$

expanding Equation A-26

$$w_1 = \frac{-\lambda}{4\pi} \int_{a_2}^{a_1} \frac{a + \lambda y'}{g}\, da$$

$$+ \frac{\lambda - y'}{4\pi} \int_{a_2}^{a_1} \frac{1}{g}\, da$$

$$+ \frac{(\lambda + 1)^{(1/2)} (x' + \lambda y')}{4\pi} \int_{a_2}^{a_1} \frac{f^2 e' - e f f'}{f^2 (f^2 + e^2)^{(1/2)}}$$

$$+ \frac{z' (\lambda + 1)^{(1/2)}}{4\pi} \int_{a_2}^{a_1} \frac{q g z' g' z' - g^2 z' q'}{(z'^2 g^2 + q^2) g z'} \qquad \text{[A-27]}$$

integrating Equation A-27

$$w_1 = \frac{-1}{4\pi} F2 + \frac{(\lambda - y')}{4\pi} G2$$

$$- \frac{(\lambda^2 + 1)^{(1/2)} (x' + \lambda y')}{4\pi} G1$$

$$+ \frac{z' \left(\lambda^2 + 1 \right)^{(1/2)}}{4\pi} F1 \qquad \text{[A-28]}$$

where

$$F2 = \left[q \right]_{a2}^{a1} \qquad \text{[A-29]}$$

This concluded the integration of the panel velocities. Next is the quadratic side vorticity. Equation A-30 give the integral for the bound side vortex of quadratic strength.

$$V_{qs} = \frac{-r'}{4\pi} \int_{\xi=0}^{\xi=1} \frac{((1/2)\,\xi^2)}{\left(r'^2 + (x'-\xi)^2\right)^{(3/2)}}\, d\xi \qquad \text{[A-30]}$$

using the following substitutions for this integration

$$a = x' - \xi$$

$$d\,a = a' = -d\,\xi$$

$$a_1 = x' - 1$$

$$a_2 = x'$$

then

$$V_{qs} = \frac{r'}{4\pi} \int_{a_2}^{a_1} \frac{(1/2)x'^2 + a(-x') + (1/2)\,a^2}{\left((r'^2 + a^2)\right)^{(3/2)}}\, d\,a \qquad \text{[A-31]}$$

noting that (Reference 24) .

$$\int \frac{1}{(x^2 + a)^{(3/2)}}\,dx = \frac{x}{a\,(x^2 + a)^{(1/2)}}$$

$$\int \frac{x}{(x^2 + a)^{(3/2)}}\,dx = \frac{-1}{(x^2 + a)^{(1/2)}}$$

$$\int \frac{x^2}{(x^2 + a)^{(3/2)}}\,dx = \frac{-x}{(x^2 + a)^{(1/2)}} + \sinh^{-1}\left(\frac{x}{a}\right)$$

then

$$V_{qs} = \frac{r'}{2}\left[\frac{x'^2}{r'^2}\frac{(x' - \xi)}{\left(r'^2 + (x' - \xi)^2\right)^{(1/2)}}\right]_{\xi=0}^{\xi=1} + \frac{r'}{2}\,G2 \qquad \text{[A-32]}$$

The u', v', and w' components are:

$$u' = 0 \qquad \text{[A-33]}$$

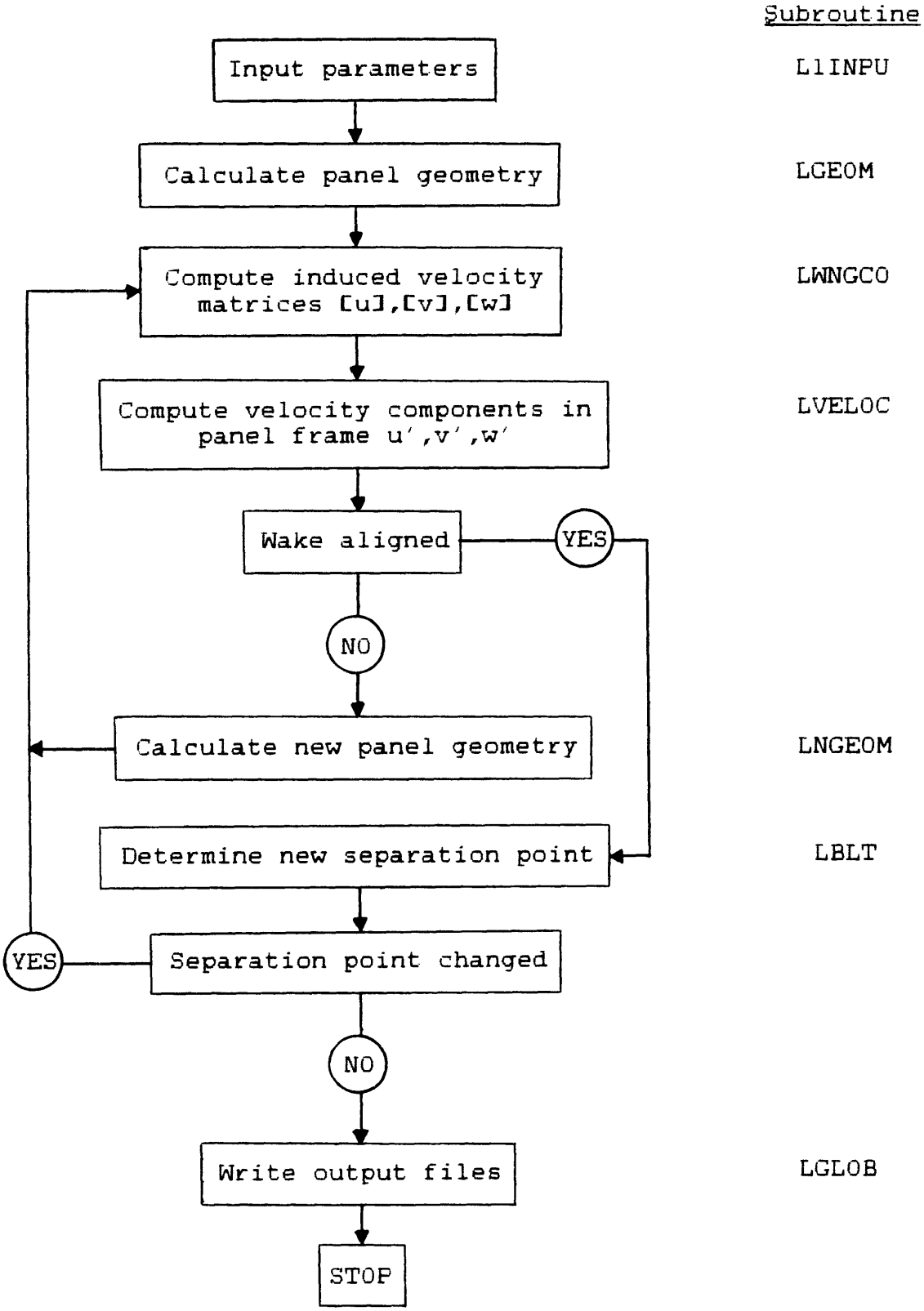
$$v' = -\frac{z'}{r'}\,V_{qs} \qquad \text{[A-34]}$$

$$w' = \frac{y'}{r'}\,V_{qs} \qquad \text{[A-35]}$$

APPENDIX B

CODE LFLY

FLOW CHART OF CODE LFLY



SAMPLE INPUT

```
004                ! run number xxx this file is lfly.xxx
clt/clfa vs nst
0                  ! reynolds number 0== attached flow
2                  ! nw  number of surfaces (wings + tunnels)
0012               ! stype NACA 4 digit airfoil  ! wing 1
0,0                ! x0,z0
10                 ! angle of attack
0                  ! dihedral
0                  ! oml - sweep angle of leading edge
0                  ! om2 - sweep angle of trailing edge
1.0                ! wing root chord
2.5                ! s (b/2)
1                  ! ns       spanwise panels
5                  ! nc       chordwise panels
1                  ! ncwak    chordwise wake panels
0.5                ! wake length in units of chord
-10                ! stype  -10 == tunnel
-4.5,0             ! x0,z0   offset
10.                ! tunnel length
-1.                ! gbh (-1 = no ground board )
7.5                ! maximum tunnel ellipse height
5.625              ! maximun tunnel ellipse width
1                  ! nst    tunnel panels per 1/4 ring
4                  ! nct    number of lengthwise tunnel panels
```

SAMPLE OUTPUT

File LFLY3F.004 09-OCT-87 04:05:14 04:06:33
Title clt/clfa vs ns tunnel
Number of wings 2
Iteration 2 solution converged
Reynolds # (M) 0.00

Wing number, type 1 12

alpha	ns	nc	ncwake	AR	taper	sweep(1/4c)	wake length
10.00	1	5	1	5.00	1.00	0.00	0.50

x0	z0	dehedral	omega(le)	omega(te)	c	b/2	area
0.0000	0.0000	0.00	0.00	0.00	1.00	2.50	5.00

CL = 0.6751 CD = 0.0636
CLgamma 0.7453

ins	Capgam	Clg(local)	CLgfrac	cl	cd	clfrac	cdfrac
1	0.3726	0.7453	0.7453	0.6751	0.0636	0.6751	0.0636

iul	ins	inc	ip	gle	gte	ul	vl	wl	cp	lamda	theta	chi	c	b
1	1	1	1	1.4075	1.7538	1.6718	0.0007	0.0000	-1.7948	0.0000	0.4493	0.0000	0.0603	2.5000
1	1	2	2	1.7538	1.4217	1.5419	0.0002	0.0000	-1.3774	0.0000	-0.0236	0.0000	0.1437	2.5000
1	1	3	3	1.4217	1.2415	1.3123	-0.0003	0.0000	-0.7221	0.0000	-0.1712	0.0000	0.2212	2.5000
1	1	4	4	1.2415	1.0773	1.1437	-0.0009	0.0000	-0.3080	0.0000	-0.2463	0.0000	0.2795	2.5000
1	1	5	5	1.0773	0.0000	0.9172	-0.0011	0.0426	0.1587	0.0000	-0.2913	0.0000	0.3111	2.5000
1	1	6	6	0.0000	0.0000	0.9621	-0.0017	0.0028	0.0743	0.0000	-0.0839	0.0000	0.5000	2.5000
2	1	1	7	1.4075	-0.5708	-0.1722	0.0012	0.0000	0.9703	0.0000	-0.7984	0.0000	0.0603	2.5000
2	1	2	8	-0.5708	-0.8915	0.7318	0.0015	0.0000	0.4645	0.0000	-0.3254	0.0000	0.1437	2.5000
2	1	3	9	-0.8915	-0.9771	0.9230	0.0016	0.0000	0.1480	0.0000	-0.1778	0.0000	0.2212	2.5000
2	1	4	10	-0.9771	-1.0631	0.9990	0.0011	0.0000	0.0021	0.0000	-0.1028	0.0000	0.2795	2.5000
2	1	5	11	-1.0631	0.0000	0.9172	-0.0001	0.0671	0.1587	0.0000	-0.0578	0.0000	0.3111	2.5000
2	1	6	12	0.0000	0.0000	0.9620	-0.0016	0.0028	0.0746	0.0000	-0.0837	0.0000	0.5000	2.5000

Wing number, type 2 -10

alpha	ns	nc	ncwake	AR	taper	sweep(1/4c)	wake length
0.00	1	4	0	0.75	1.00	0.00	0.00

x0	z0	dehedral	omega(le)	omega(te)	c	b/2	area
-4.5000	0.0000	0.00	0.00	0.00	10.00	3.75	75.00

CL = -0.0563 CD = 0.0000
CLgamma -0.0674

ins	Capgam	Clg(local)	CLgfrac	cl	cd	clfrac	cdfrac
1	-0.3596	-0.0674	-0.0674	-0.0563	0.0000	-0.0563	0.0000

iul	ins	inc	ip	gle	gte	ul	vl	wl	cp	lamda	theta	chi	c	b
1	1	1	13	-0.0339	-0.0275	0.9788	0.0029	0.0000	0.0420	0.0000	0.0000	1.0584	2.5000	7.1709
1	1	2	14	-0.0275	-0.0415	0.9696	0.0060	0.0000	0.0599	0.0000	0.0000	1.0584	2.5000	7.1709
1	1	3	15	-0.0415	-0.0136	0.9758	0.0048	0.0000	0.0478	0.0000	0.0000	1.0584	2.5000	7.1709
1	1	4	16	-0.0136	0.0000	0.9928	0.0068	0.0000	0.0143	0.0000	0.0000	1.0584	2.5000	7.1709
2	1	1	17	0.0031	-0.0111	1.0021	-0.0032	0.0000	-0.0042	0.0000	0.0000	-1.0584	2.5000	7.1709

2	1	2	18	-0.0111	-0.0303	1.0189	-0.0068	0.0000	-0.0381	0.0000	0.0000	-1.0584	2.5000	7.1709
2	1	3	19	-0.0303	-0.0045	1.0151	-0.0030	0.0000	-0.0304	0.0000	0.0000	-1.0584	2.5000	7.1709
2	1	4	20	-0.0045	0.0000	1.0017	-0.0058	0.0000	-0.0034	0.0000	0.0000	-1.0584	2.5000	7.1709

CODE LISTING

```
PDP-11 FORTRAN-77 V5.0-0        15:03:20    13-Nov-87          Page 1
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0001          program lfly

        c       [20,64]lfly.ftn
        c       separation added by modify [20,60]lfly1,2,3
        c       and adding a boundary layer analysis
        c       wind tunnel wall interference program
        c       3-d separated flow inside a wind tunnel
        c       for input see [20,64]linpul
        c

0002          implicit character*200 (a-z)

        c     global variables needed throughout
0003          logical      finish       ! true if the solution has converged
0004          character*8  stime        ! time of start
0005          character*8  ctime      ! current time

0006          include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0007 *          integer    pmax
0008 *          parameter (pmax = 90) ! max number of panels
0009 *          integer    nwmax
0010 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0011 *          integer    nsmax
0012 *          parameter (nsmax=2)  ! max number of spanwise stations
0013 *          integer    ncmax
0014 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0015 *          integer    itmax
0016 *          parameter (itmax=5) ! max number of total iterations
0017          include 'lwingl.com'
     *  c       file [20,64]lwingl.com
     *
0018 *          integer  nw,                ! number of surfaces
     *          1                           ! wing + wake(s), & tunnel
     *          2        ign,               ! number of gammas
     *          3        ipn                ! number of panels
0019 *          real     la1(nwmax),
     *          1        la2(nwmax),
     *          2        sw(nwmax),         ! half span = b/2
     *          2        cwak(nwmax),       ! length of wake
     *          3        cw(nwmax),         ! chord of wing
     *          4        x0(nwmax),
     *          5        z0(nwmax),
     *          6        alpha(nwmax),
     *          7        dihed(nwmax),
     *          8        eh,                ! tunnel ellipse height
     *          9        gbh,               ! tunnel ground board height
     *          1        reyn               ! reynolds number e-6
0020 *          integer  stype(nwmax)       ! surface type wing wake or tunnel
     *  c                                     >0 NACA 4 digit airfoil
     *  c                                     =0 NACA 64A005
     *  c                                     = -10 tunnel
     *  c
0021 *          integer  nc(nwmax),         ! wing chordwise panels
```

```
PDP-11 FORTRAN-77 V5.0-0        15:03:20    13-Nov-87          Page 2
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

     *          2        nct(nwmax),       ! total chordwise panels
     *          2        ns(nwmax),        ! spanwise panels
     *          3       !bc(nwmax),        ! boundary condition
     *          4        nsep(nwmax,nsmax)! separation pt
     *  c                separates at nsep panels upstream of trailing edge
0022 *          integer  itws,             ! wake shape iterations
     *          6        itsp,             ! separation pt iterations
     *          7        itto              ! total iterations
     *
     *
0023 *          common /lwingl/ nw, ign, ipn, la1, la2, sw, cwak, cw, x0, z0,
     *          1               alpha, dihed, eh, gbh, reyn, stype,
     *          2               nc, nct, ns, nsep, itws, itsp, itto
     *
0024           include 'lchar.com'
     *  c       file [20,64]lchar.com
     *
0025 *          character*40  title
0026 *          character*4  run
     *
0027 *          common /lchar/  title,run
     *
0028           include 'lhist.com'
     *  c       file [20,64]lhist.com
     *  c       Used to keep information for convergence history of method
     *
     *
0029 *          real    cl(nwmax,itmax) !cl vs iteration history
     *
0030 *          common /lhist/ cl
     *
     *
0031           real      clip(pmax),cdip(pmax)



0032           call time(stime)

0033           type*, ' lfly started at   ',stime

0034           call lflyl         ! create geometry for wing(s) and tunnel

0035           call time(ctime)
0036           type*, ' lflyl finished at ',ctime

0037           type *, ' run number ',run(1:3)
0038           type *, title(1:40)

0039           write(3,*) ' run number ',run(1:3)
0040           write(3,*) title(1:40)

0041           itto = 1  ! total iterations
0042           itws = 1  ! iteration wake shape for this sep pt
0043           itsp = 1  ! iteration separation point
```

```
PDP-11 FORTRAN-77 V5.0-0        15:03:20    13-Nov-87          Page 3
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0044    201     call lfly2      ! compute matrix of velocity coefficients

0045            call time(ctime)
0046            type*, ' lfly2 finished at ',ctime

        c       solve flow field and check for convergence
0047            call lfly3(clip,cdip,finish,stime)

0048            call time(ctime)
0049            type*, ' lfly3 finished at ',ctime

0050            if ((.not.finish).and.itto.le.itmax) goto 201  ! iterate if needed

0051          end


PROGRAM SECTIONS

 Name        Size                 Attributes

$CODE1  000610   196              RW,I,CON,LCL
$PDATA  000154    54              RW,D,CON,LCL
$VARS   001342   369              RW,D,CON,LCL
LWING1  000170    60              RW,D,OVR,GBL
LCHAR   000054    22              RW,D,OVR,GBL
LHIST   000050    20              RW,D,OVR,GBL


Total Space Allocated = 002642   721

No FPP Instructions Generated
```

```
0001          subroutine lflyl

        c       file [20,64]lflyl.ftn
        c       input : wing geometry - see llinput
        c       output: writes the panel configuration to files lflylf.xxx and
        c               lflylu.xxx which are formatted and unformatted respectively


0002          implicit character*200 (a-z)

0003          include 'lfly.inc'      ! gives maximum number of panels
     *  c       file [20,64]lfly.inc
0004 *          integer    pmax
0005 *          parameter (pmax = 90) ! max number of panels
0006 *          integer    nwmax
0007 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0008 *          integer    nsmax
0009 *          parameter (nsmax=2)  ! max number of spanwise stations
0010 *          integer    ncmax
0011 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0012 *          integer    itmax
0013 *          parameter (itmax=5) ! max number of total iterations
0014          include 'lchar.com'     ! charater data
     *  c       file [20,64]lchar.com
     *
0015 *          character*40  title
0016 *          character*4  run
     *
0017 *          common /lchar/ title,run
     *
0018          include 'lwingl.com'    ! wing geometry
     *  c       file [20,64]lwingl.com
     *
0019 *          integer  nw,                ! number of surfaces
     *          1                           ! wing + wake(s), & tunnel
     *          2        ign,               ! number of gammas
     *          3        ipn                ! number of panels
0020 *          real     la1(nwmax),
     *          1        la2(nwmax),
     *          2        sw(nwmax),         ! half span = b/2
     *          2        cwak(nwmax),       ! length of wake
     *          3        cw(nwmax),         ! chord of wing
     *          4        x0(nwmax),
     *          5        z0(nwmax),
     *          6        alpha(nwmax),
     *          7        dihed(nwmax),
     *          8        eh,                ! tunnel ellipse height
     *          9        gbh,               ! tunnel ground board height
     *          1        reyn               ! reynolds number e-6
0021 *          integer  stype(nwmax)       ! surface type wing wake or tunnel
     *  c                                     >0 NACA 4 digit airfoil
     *  c                                     =0 NACA 64A005
     *  c                                     = -10 tunnel
```

```
PDP-11 FORTRAN-77 V5.0-0        15:03:27    13-Nov-87          Page 5
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

      *  c
0022  *           integer  nc(nwmax),        ! wing chordwise panels
      *           2        nct(nwmax),       ! total chordwise panels
      *           2        ns(nwmax),        ! spanwise panels
      *           3       !bc(nwmax),        ! boundary condition
      *           4        nsep(nwmax,nsmax)! separation pt
      *  c                 separates at nsep panels upstream of trailing edge
0023  *           integer  itws,             ! wake shape iterations
      *           6        itsp,             ! separation pt iterations
      *           7        itto              ! total iterations
      *
      *
0024  *           common /lwing1/ nw, ign, ipn, la1, la2, sw, cwak, cw, x0, z0,
      *           1               alpha, dihed, eh, gbh, reyn, stype,
      *           2               nc, nct, ns, nsep, itws, itsp, itto
      *
0025            include 'lwing2.var'    ! panel geometry
      *  c       file 'lwing2.var'
      *  c       changed from common block to allow use of virtual memory
      *
0026  *           integer iga,
      *           1       iul,
      *           2       inc,
      *           3       iwng
0027  *           real    la,
      *           1       theta,
      *           2       chi,
      *           3       c,
      *           4       b,
      *           5       xp, yp, zp,
      *           6       xc, yc, zc
      *
0028  *         virtual    iga(pmax), iul(pmax), inc(pmax), iwng(pmax),
      *        1           la(pmax), theta(pmax), chi(pmax), c(pmax), b(pmax),
      *        2           xp(pmax), yp(pmax), zp(pmax),
      *        3           xc(pmax), yc(pmax), zc(pmax)
      *

0029            character*10      fname


0030            call llinpu

0031            fname(8:10) = run(1:3)

        d       fname(1:7) = 'lflylf.'  ! formatted - can be typed out
        d       open(unit=1,name=fname,status='new',dispose='keep')

0032            fname(1:7) = 'lflylu.'  ! unformatted - input for lfly2

0033            open(unit=2,name=fname,form='unformatted',
               1    status='new',dispose='keep')

0034            call lgeom
```

```
        d       close(1)
0035            close(2)

0036            end
```

PROGRAM SECTIONS

Name	Size		Attributes
$CODE1	000146	51	RW,I,CON,LCL
$PDATA	000056	23	RW,D,CON,LCL
$IDATA	000170	60	RW,D,CON,LCL
$VARS	000012	5	RW,D,CON,LCL
LCHAR	000054	22	RW,D,OVR,GBL
LWING1	000170	60	RW,D,OVR,GBL

Total Space Allocated = 000672 221

Total Virtual Array Storage = 78

No FPP Instructions Generated

```
PDP-11 FORTRAN-77 V5.0-0        11:30:52    17-Nov-87          Page 6a
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0001            subroutine llinpu

        c       file [20,64]llinpu.ftn
        c       find run number in file lfly.num and read input file lfly.xxx
        c       fill common block lwingl

0002            implicit character*200 (a-z)

0003            include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0004 *          integer    pmax
0005 *          parameter (pmax = 90) ! max number of panels
0006 *          integer    nwmax
0007 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0008 *          integer    nsmax
0009 *          parameter (nsmax=2)  ! max number of spanwise stations
0010 *          integer    ncmax
0011 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0012 *          integer    itmax
0013 *          parameter (itmax=5) ! max number of total iterations
0014            include 'lchar.com'
     *  c       file [20,64]lchar.com
     *
0015 *          character*40  title
0016 *          character*4  run
     *
0017 *          common /lchar/  title,run
     *
0018            include 'lwingl.com'
     *  c       file [20,64]lwingl.com
     *
0019 *          integer  nw,                ! number of surfaces
     *          1                           ! wing + wake(s), & tunnel
     *          2        ign,               ! number of gammas
     *          3        ipn                ! number of panels
0020 *          real     la1(nwmax),
     *          1        la2(nwmax),
     *          2        sw(nwmax),         ! half span = b/2
     *          2        cwak(nwmax),       ! length of wake
     *          3        cw(nwmax),         ! chord of wing
     *          4        x0(nwmax),
     *          5        z0(nwmax),
     *          6        alpha(nwmax),
     *          7        dihed(nwmax),
     *          8        eh,                ! tunnel ellipse height
     *          9        gbh,               ! tunnel ground board height
     *          1        reyn               ! reynolds number e-6
0021 *          integer  stype(nwmax)       ! surface type wing wake or tunnel
     *  c                                     >0 NACA 4 digit airfoil
     *  c                                     =0 NACA 64A005
     *  c                                     = -10 tunnel
     *  c
0022 *          integer  nc(nwmax),         ! wing chordwise panels
     *          2        nct(nwmax),        ! total chordwise panels
```

```
PDP-11 FORTRAN-77 V5.0-0        11:30:52    17-Nov-87          Page 6b
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

     *          2        ns(nwmax),        ! spanwise panels
     *          3       !bc(nwmax),        ! boundary condition
     *          4        nsep(nwmax,nsmax)! separation pt
     *  c                separates at nsep panels upstream of trailing edge
0023 *          integer  itws,             ! wake shape iterations
     *          6        itsp,             ! separation pt iterations
     *          7        itto              ! total iterations
     *
     *
0024 *          common /lwingl/ nw, ign, ipn, lal, la2, sw, cwak, cw, x0, z0,
     *          1               alpha, dihed, eh, gbh, reyn, stype,
     *          2               nc, nct, ns, nsep, itws, itsp, itto
     *

0025            real omega1,omega2
0026            real pi

0027            integer iunit,
                1       inw,
                2       nrun,
                3       ins

0028            character*8 fname
0029            character*3 run1

0030            pi  = acos(-1.0)

0031            iunit = 1 ! file

0032            open(unit=iunit,name='lfly.num',status='old',err=901)
0033            read(iunit,*,err=901) nrun
0034            close(iunit)

        d       type *, '  nrun = ',nrun

        d       if (nrun.lt.0.or.nrun.gt.999) then
        cd901     type *, ' input run number 000 to 999  '
        d         type *, ' input run number 000 to 999  '
        d         accept *, nrun
        d       end if

0035            write (run(1:3), '(i3)' ) nrun

0036            if (run(1:1).eq.' ') run(1:1) = '0'
0037            if (run(2:2).eq.' ') run(2:2) = '0'

        d       type '(2a)',  ' run number =  ',run(1:3)

0038            fname(1:5) = 'lfly.'
0039            fname(6:8) = run(1:3)

0040            open (unit=iunit,name=fname,status='old',err=901)

0041            read(iunit,'(a)',err=901) run1(1:3)
```

```
PDP-11 FORTRAN-77 V5.0-0        11:30:52    17-Nov-87          Page 6c
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0042            if (run(1:3).ne.run1(1:3)) then
0043              type *, ' file number does not match file name
0044              type *, ' run number = ', run1(1:3)
0045              type *, ' file name = ', fname(1:8)
0046              call exit
0047            end if

        c
        c       Consider changing the input parameters to:
        c         aspect ratio
        c         taper
        c         sweep of 1/4 chord line
        c

0048            read(iunit,'(a)',err=902) title
        d       type *,' title = ',title(1:40)


        c       base reynolds number on wing 1 chord
0049            read(iunit,*,err=902) reyn
        d       type *,' reynolds number = ',reyn,' X 10^6 '
0050            reyn = reyn * 10**6

0051            read(iunit,*,err=902) nw
        d       type *,' number of surfaces  = ',nw

        c
        c       a surface is either a wing and wake or tunnel
        c       a wing with wake counts as one surface
        c
        c


0052            do 101 inw = 1,nw
        d         type *,' input stype for surface number ',inw
0053              read(iunit,*,err=902) stype(inw)

        d         type *, ' surface type ',stype(inw)
        c         stype > 0 then NACA 4 digit airfoil number
        c         stype = 0 then NACA 64A005  airfoil
        c         stype >-10,<0 reserved for special wings
        c         stype =-10 then tunnel


0054              if (stype(inw).gt.-1) then ! normal wing

        d           type *, ' x0 = down stream offset of leading edge '
        d           type *, ' z0 = elevation offset of leading edge '

0055                read(iunit,*,err=902)  x0(inw),z0(inw)
        d           type *,' x0,z0 of inboard leading edge = ',x0(inw),z0(inw)
        c           the location of the firt wing should be 0,0


0056                read(iunit,*,err=902)  alpha(inw)
```

```
PDP-11 FORTRAN-77 V5.0-0        11:30:52    17-Nov-87          Page 6d
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

      d           type *,' angle of attack (deg) = ',alpha(inw)
0057              alpha(inw) = alpha(inw) *pi/180

0058              read(iunit,*,err=902)  dihed(inw)
      d           type *,' dihedral angle (deg) = ',dihed(inw)
0059              dihed(inw) = dihed(inw) *pi/180
      c           assume small angle cos(dihed) ~ 1 for wing shape
      c           and surface continuity at y=0

0060              read(iunit,*,err=902) omega1
      d           type *,' leading edge sweep angle (deg) = ',omega1
0061              la1(inw) = tan(omega1*pi/180)

0062              read(iunit,*,err=902) omega2
      d           type *,' trailing edge sweep angle (deg) = ',omega2
0063              la2(inw) = tan(omega2*pi/180)

0064              read(iunit,*,err=902) cw(inw)
      d           type *, ' inboard chord of wing = ', cw(inw)
      c           for 2-D approximation use 1 on first wing
      c           variable for 3-D

0065              read(iunit,*,err=902) sw(inw)
      d           type *, ' half span of wing = ',sw(inw)
      c           for 3_D sw of the firt wing should be 1.0
      c           for 2_D use 100

0066              read(iunit,*,err=902) ns(inw)
      d           type *, ' number of spanwise wing panels = ', ns(inw)

      c           initially assume no separation
0067              do 111 ins = 1,ns(inw)
0068                nsep(inw,ins) = 0
0069    111       continue

0070              read(iunit,*,err=902) nc(inw)
      d           type *, ' number of chordwise wing panels = ',nc(inw)

0071              read(iunit,*,err=902) nct(inw)
      d           type *, ' number of wake chordwise panels = ',nct(inw)
0072              nct(inw) = nc(inw) + nct(inw)
      c           total number of panels along chord nc + nc_wake


0073              read(iunit,*,err=902) cwak(inw)
      d           type *, ' wake length in units of chord = ', cw(inw)

      c           units of chord of wing that this is the wake for
      c           convert to distance
0074              cwak(inw) = cwak(inw)*cw(inw)

      c           read(iunit,*,err=902) bc(inw)
      cd          type *, ' boundary condition number = ',bc(inw)
      c           use 5 for the pressent time
```

```
0075              else if (stype(inw).eq.-10) then ! tunnel

0076                alpha(inw) = 0
0077                dihed(inw) = 0
0078                lal(inw) =   0
0079                la2(inw) =   0

0080                read(iunit,*,err=902)  x0(inw),z0(inw)
        d           type *,' x0,z0 of center, start of tunnel = ',x0(inw),z0(inw)

0081                read(iunit,*,err=902) cw(inw)
        d           type *, ' tunnel length = ', cw(inw)

0082                read(iunit,*,err=902) gbh
        d           type *, ' negitive ground board height for no ground board and
        d           type *, ' wing in normal configuration '
        d           type *, ' 0 <= gbh <= 1.0 for test with wing vertical '
        d           type *, ' ground board heigth (frac. of height) = ', gbh

0083                if (gbh.lt.0) then ! no ground board
0084                  gbh = 0.5

0085                  read(iunit,*,err=902) sw(inw)
        d             type *, ' maximum tunnel ellipse width = ',sw(inw)
0086                  sw(inw) = sw(inw)/2.

0087                  read(iunit,*,err=902) eh
        d             type *, ' maximun tunnel ellipse height = ',eh

0088                else  ! ground board
0089                  read(iunit,*,err=902) eh
        d             type *, ' maximum tunnel ellipse width = ',eh

0090                  read(iunit,*,err=902) sw(inw)
        d             type *, ' maximun tunnel ellipse height = ',sw(inw)
0091                  sw(inw) = sw(inw)/2.
0092                end if

0093                read(iunit,*,err=902) ns(inw)
        d           type *, ' half number of spanwise tunnel panels = ', ns(inw)

0094                read(iunit,*,err=902) nc(inw)
        d           type *, ' number of lengthwise panels = ',nc(inw)
0095                nct(inw) = nc(inw)

        c           bc(inw) = 0
        cd          type *, ' boundary condition number = ',bc(inw)

0096              end if

0097    101     continue
0098            close (iunit)

0099            itws = 0
```

```
PDP-11 FORTRAN-77 V5.0-0        15:03:33    13-Nov-87          Page 7
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0001          subroutine lgeom

        c       panel wing and find panel parameters 'lwing2.var'

0002          implicit character*200 (a-z)

0003          include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0004 *          integer    pmax
0005 *          parameter (pmax = 90) ! max number of panels
0006 *          integer    nwmax
0007 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0008 *          integer    nsmax
0009 *          parameter (nsmax=2)  ! max number of spanwise stations
0010 *          integer    ncmax
0011 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0012 *          integer    itmax
0013 *          parameter (itmax=5) ! max number of total iterations
0014          include 'lwing1.com'
     *  c       file [20,64]lwing1.com
     *
0015 *          integer  nw,                ! number of surfaces
     *          1                           ! wing + wake(s), & tunnel
     *          2        ign,               ! number of gammas
     *          3        ipn                ! number of panels
0016 *          real     la1(nwmax),
     *          1        la2(nwmax),
     *          2        sw(nwmax),         ! half span = b/2
     *          2        cwak(nwmax),       ! length of wake
     *          3        cw(nwmax),         ! chord of wing
     *          4        x0(nwmax),
     *          5        z0(nwmax),
     *          6        alpha(nwmax),
     *          7        dihed(nwmax),
     *          8        eh,                ! tunnel ellipse height
     *          9        gbh,               ! tunnel ground board height
     *          1        reyn               ! reynolds number e-6
0017 *          integer  stype(nwmax)       ! surface type wing wake or tunnel
     *  c                                     >0 NACA 4 digit airfoil
     *  c                                     =0 NACA 64A005
     *  c                                     = -10 tunnel
     *  c
0018 *          integer  nc(nwmax),         ! wing chordwise panels
     *          2        nct(nwmax),        ! total chordwise panels
     *          2        ns(nwmax),         ! spanwise panels
     *          3       !bc(nwmax),         ! boundary condition
     *          4        nsep(nwmax,nsmax)! separation pt
     *  c                separates at nsep panels upstream of trailing edge
0019 *          integer  itws,              ! wake shape iterations
     *          6        itsp,              ! separation pt iterations
     *          7        itto               ! total iterations
     *
     *
0020 *          common /lwing1/ nw, ign, ipn, la1, la2, sw, cwak, cw, x0, z0,
```

```
PDP-11 FORTRAN-77 V5.0-0        15:03:33    13-Nov-87          Page 8
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

     *          1               alpha, dihed, eh, gbh, reyn, stype,
     *          2               nc, nct, ns, nsep, itws, itsp, itto
     *
0021           include 'lwing2.var'
     *  c       file 'lwing2.var'
     *  c       changed from common block to allow use of virtual memory
     *
0022 *          integer iga,
     *          1       iul,
     *          2       inc,
     *          3       iwng
0023 *          real    la,
     *          1       theta,
     *          2       chi,
     *          3       c,
     *          4       b,
     *          5       xp, yp, zp,
     *          6       xc, yc, zc
     *
0024 *         virtual    iga(pmax), iul(pmax), inc(pmax), iwng(pmax),
     *        1           la(pmax), theta(pmax), chi(pmax), c(pmax), b(pmax),
     *        2           xp(pmax), yp(pmax), zp(pmax),
     *        3           xc(pmax), yc(pmax), zc(pmax)
     *

0025           real      x1(4),y1(4),z1(4)  ! wing ref frame
0026           real      x2(4),y2(4),z2(4)  ! global ref frame
0027           real      x3(4),y3(4),z3(4)  ! panel ref frame

0028           real    a,          ! taper parameter lal - la2
                 1       bli,blo,    ! y for inboard/outboard edges of a panel
                 2       cli,clo     ! wing chord for inboard/outboard edges of a pane
0029           integer incl,       ! chordwise paneling counter, wing
                 1       inc2,       ! chordwise paneling counter, wake
                 1       ins,        ! spanwise paneling counter
                 2       inw,        ! wing counter
                 3       iunit,
                 4       iull,       ! surface counter
                 5       ipt,ip

0030           real      sth,cth,cch,sch
0031           real      xt,yt,zt
0032           real    xp3
0033           real      lwing       ! wing cross section shape function
0034           real    pi

0035           real    xple,     ! x fraction of chord for panel leading edge
                 1       xpte      ! x fraction of chord for panel trailing edge

0036           real    xte(nsmax,2),
                 1       zte(nsmax,2)


       d       integer j
```

```
PDP-11 FORTRAN-77 V5.0-0        15:03:33    13-Nov-87          Page 9
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

       d     close(3) ! debug

0037          pi = acos(-1.0)
0038          ip = 0                          !  panel counter
0039          iunit = 1

       d       write(iunit,'(3a)')
       d       1   '   ip  iga iul inc iwng      la     theta      chi        c    ,
       d       2   '         b        xp       yp       zp        xc       yc       zc'

0040          do 101 inw = 1,nw
       cd       type *,' wing ',inw

0041            if (stype(inw).eq.-10) then ! surface is the tunnel
       c          write panel discription of tunnel to lflylu.xxx
0042              call ltun ( ip, inw,
                  2                iga, iul, inc, iwng,
                  2                la, theta, chi, c, b,
                  2                xp, yp, zp, xc, yc, zc                  )


0043            else ! normal wing + wake
0044              a = la1(inw) - la2(inw)
       cd         type *,' a ',a
0045              do 111 iul1 = 1,2                          ! upper/lower surface
0046                do 121 ins = 1,ns(inw)                   ! span divisions
       c              wing ref frame
0047                  bli = (ins-1.)/ns(inw)*sw(inw)         ! y offset-  b inboard local
0048                  cli = cw(inw) - a*bli                  ! wing chord inboard local

0049                  blo = (ins*1.)/ns(inw)*sw(inw)
0050                  clo = cw(inw) - a*blo                   ! chord outboard local
       cd             type *, ' bli,cli,blo,clo ',bli,cli,blo,clo

0051                  do 131 incl = 1,nct(inw)

       cd     type *, ' incl ',incl
0052          ip = ip + 1

0053          if (incl.le.nc(inw)) then ! on wing

       c        find corner points for a swept tapered panel in wing ref frame

       c        xple  = (0.5-0.5*cos((incl-1.)/nc(inw)*pi))
       c        xpte  = (0.5-0.5*cos((incl*1.)/nc(inw)*pi))

0054            xple  = (1.0-    cos((incl-1.)/nc(inw)*pi/2))
0055            xpte  = (1.0-    cos((incl*1.)/nc(inw)*pi/2))

0056            xl(1) = la1(inw)*bli + xple*cli
0057            yl(1) = bli                                        *cos(dihed(inw))
0058            zl(1) = cli*lwing( xple, iul1, stype(inw) ) +bli*sin(dihed(inw))

0059            xl(2) = la1(inw)*bli + xpte*cli
0060            yl(2) = bli                                        *cos(dihed(inw))
```

```
0061            zl(2) = cli*lwing( xpte, iull, stype(inw) ) +bli*sin(dihed(inw))

0062            xl(3) = lal(inw)*blo + xple*clo
0063            yl(3) = blo                                        *cos(dihed(inw))
0064            zl(3) = clo*lwing( xple, iull, stype(inw) ) +blo*sin(dihed(inw))

0065            xl(4) = lal(inw)*blo + xpte*clo
0066            yl(4) = blo                                        *cos(dihed(inw))
0067            zl(4) = clo*lwing( xpte, iull, stype(inw) ) +blo*sin(dihed(inw))


        c       add angle of attack and translate
0068            sth = sin(-alpha(inw))
0069            cth = cos(-alpha(inw))

        cd      type *, ' sth,cth  ',sth,cth

        c       x,y,z => global frame

0070            x2(1) = x0(inw) +  xl(1)*cth          - zl(1)*sth
0071            y2(1) =                          yl(1)
0072            z2(1) = z0(inw) +  xl(1)*sth          + zl(1)*cth

0073            x2(2) = x0(inw) +  xl(2)*cth          - zl(2)*sth
0074            y2(2) =                          yl(2)
0075            z2(2) = z0(inw) +  xl(2)*sth          + zl(2)*cth

0076            if (incl.eq.nc(inw)) then ! trailing edge of wing
0077              xte(ins,iull) = x2(2)
0078              zte(ins,iull) = z2(2)
0079            end if

0080            x2(3) = x0(inw) +  xl(3)*cth          - zl(3)*sth
0081            y2(3) =                          yl(3)
0082            z2(3) = z0(inw) +  xl(3)*sth          + zl(3)*cth

0083            x2(4) = x0(inw) +  xl(4)*cth          - zl(4)*sth
0084            y2(4) =                          yl(4)
0085            z2(4) = z0(inw) +  xl(4)*sth          + zl(4)*cth

0086          else ! on wake of wing
        c       find corner points for a swept wake panel in wing ref frame

0087            inc2 = incl - nc(inw)

0088            xple  = (inc2-1.)/(nct(inw) - nc(inw))
0089            xpte  = (inc2*1.)/(nct(inw) - nc(inw))

        c       modifying the wing cross section funcion to include
        c       the initial shape of wakes me be worth it

0090            xl(1) = la2(inw)*bli + xple*cwak(inw)
0091            yl(1) = bli                                  *cos(dihed(inw))
0092            zl(1) = bli*sin(dihed(inw))
```

```
PDP-11 FORTRAN-77 V5.0-0        15:03:33    13-Nov-87          Page 11
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0093            xl(2) = la2(inw)*bli + xpte*cwak(inw)
0094            yl(2) = bli                                *cos(dihed(inw))
0095            zl(2) = bli*sin(dihed(inw))

0096            xl(3) = la2(inw)*blo + xple*cwak(inw)
0097            yl(3) = blo                                *cos(dihed(inw))
0098            zl(3) = blo*sin(dihed(inw))

0099            xl(4) = la2(inw)*blo + xpte*cwak(inw)
0100            yl(4) = blo                                *cos(dihed(inw))
0101            zl(4) = blo*sin(dihed(inw))

        c       find coordinates in global reference frame
        c       rotate for angle of attack then translate by x0,z0

0102            sth = sin(-alpha(inw))
0103            cth = cos(-alpha(inw))

        cd      type *, ' sth,cth  ',sth,cth

        c       x,y,z => global frame

        c       find trailing edge for the wing i.e. gives the
        c       proper of set for the upper and then the
        c       lower surface of the wake

0104            x2(1) = xte(ins,iull) +  xl(1)*cth          - zl(1)*sth
0105            y2(1) =                              yl(1)
0106            z2(1) = zte(ins,iull) +  xl(1)*sth          + zl(1)*cth

0107            x2(2) = xte(ins,iull) +  xl(2)*cth          - zl(2)*sth
0108            y2(2) =                            yl(2)
0109            z2(2) = zte(ins,iull) +  xl(2)*sth          + zl(2)*cth

0110            x2(3) = xte(ins,iull) +  xl(3)*cth          - zl(3)*sth
0111            y2(3) =                            yl(3)
0112            z2(3) = zte(ins,iull) +  xl(3)*sth          + zl(3)*cth

0113            x2(4) = xte(ins,iull) +  xl(4)*cth          - zl(4)*sth
0114            y2(4) =                            yl(4)
0115            z2(4) = zte(ins,iull) +  xl(4)*sth          + zl(4)*cth

0116          end if


        c     determine individual panel parameters

        c         write(3,*) ' ip, x2...    '
        c         do 301 j=1,4
        c           write(3,501) ip, xl(j), yl(j), zl(j),
        c       1       xl(j)-xl(1), yl(j)-yl(1), zl(j)-zl(1)
0117    501         format(i8,10f8.4)
        c301      continue
        c         type *, '     '
        c         type *,' alpha,dihed ',  alpha(inw), dihed(inw)
```

```
        c         find coordinates in global reference frame
        c         rotate for angle of attack then translate by x0,z0



        c       write(3,*) '  ip x2...
        c       do 302 j=1,4
        c       write(3,501)  ip, x2(j), y2(j), z2(j),
        c       1       x2(j)-x2(1), y2(j)-y2(1), z2(j)-z2(1)
        c302    continue
        c       write(3,*) '


        c       average the four points onto a plain with two sides of constant y
        c       to make a usable panel
        c       call lplain
        c       this is not needed since all wings have similar forms along the
        c       span and therefore only simple curvature


        c       determine individual panel parameters

0118            if (ip.eq.1) then
0119              iga(ip) = 1
0120            else
0121              iga(ip) = iga(ip-1)+1
0122              if (incl.eq.1) iga(ip) = iga(ip) + 1
0123            end if

0124            iul(ip)   = iul1
0125            inc(ip)   = incl
0126            iwng(ip)  = inw
0127            theta(ip) = atan2( (z2(2)-z2(1)), (x2(2)-x2(1)) )

0128            chi(ip) = atan2( (z2(3)-z2(1))*cos(theta(ip))-
                  1                 (x2(3)-x2(1))*sin(theta(ip)) ,
                  1                 y2(3)-y2(1)  )

        cd      type *, ' theta(ip), chi(ip) ', theta(ip), chi(ip)

0129            sth = sin(theta(ip))
0130            cth = cos(theta(ip))
0131            sch = sin(chi(ip))
0132            cch = cos(chi(ip))

        c       type *, ' sth,cth,sch,cch ',sth,cth,sch,cch

0133            do 141 ipt = 1,4

        c         find corner points in panel system (x3,y3,z3)  (z3 == 0)
        c         translate axis and rotate into panel system

0134              xt = x2(ipt) - x2(1)
0135              yt = y2(ipt) - y2(1)
```

```
0136            zt = z2(ipt) - z2(1)

0137            x3(ipt) =   xt*cth                + zt*sth
0138            y3(ipt) = - xt*sth*sch + yt*cch   + zt*cth*sch
0139            z3(ipt) = - xt*sth*cch - yt*sch   + zt*cth*cch

0140    141     continue


        c         type *, '  ip x3...
        c       do 304 j=1,4
        c         type 501, ip, x3(j), y3(j), z3(j),
        c       1       x3(j)-x3(1), y3(j)-y3(1), z3(j)-z3(1)
0141    304     continue
        c         type *, '                '

0142          b(ip) = y3(3)

        c     approximate the swept tapered panel by a swept panel

0143          la(ip) =( x3(3)       /b(ip) +
                1        (x3(4)-x3(2))/b(ip)   )/2.

0144          c(ip) = (x3(2) + (x3(4)-x3(3)) )/2.

        c       inboard leading edge point of approximate panel in
        c       tapered panel reference frame(3)
0145            xp3   = x3(3)/2. - la(ip)*b(ip)/2.
        c       type *, ' xp3 ',xp3
        c       yp3,zp3 = 0

        c       determine the new pt 1 of the panel in global coordinates
        c       rotate, translate to global coordinates

0146          xp(ip) =  x2(1) + xp3*cth     !- yp3*sch*sth - zp3*cch*sth
0147          yp(ip) =  y2(1)               !+ yp3*cch     - zp3*sch
0148          zp(ip) =  z2(1) + xp3*sth     !+ yp3*sch*cth + zp3*cch*cth

        c       find control point of panel
0149          xc(ip) = (x2(1) + x2(2) + x2(3) + x2(4) )/4.
0150          yc(ip) = (y2(1) + y2(2) + y2(3) + y2(4) )/4.
0151          zc(ip) = (z2(1) + z2(2) + z2(3) + z2(4) )/4.

        c       write(4,*) xc(ip),yc(ip)! control point in global frame

        c       if (xc(ip)-xp(ip)-c(ip)*0.5*cth.ge.1.e-4) then
        c         type*,  ' error in control pt '
        c         type*, ip,xc(ip),xp(ip) + c(ip)*0.5*cth
        c         call exit
        c       end if


        c       true so far as tested
        c       type *, ' xc test (:=:0) ',
        c       1    (xc(ip) - ((x2(1)+x2(3))/2. + c(ip)*0.5*cth) )
```

```
        d       write(iunit,502)
        d       1
        d       2  ip, iga(ip), iul(ip), inc(ip), iwng(ip),
        d       3
        d       4  la(ip), theta(ip), chi(ip), c(ip), b(ip),
        d       5
        d       6  xp(ip), yp(ip), zp(ip), xc(ip),yc(ip),zc(ip)
0152    502     format(5i4,5f10.4,x,2(3f9.4,x) )

0153          write(2)
                1
                2  ip, iga(ip), iul(ip), inc(ip), iwng(ip),
                3
                4  la(ip), theta(ip), chi(ip), c(ip), b(ip),
                5
                6  xp(ip), yp(ip), zp(ip), xc(ip),yc(ip),zc(ip)


0154    131           continue
0155    121         continue
0156    111       continue
0157              end if
0158    101   continue
        c     close(4)
0159          end
```

PROGRAM SECTIONS

Name	Size		Attributes
$CODE1	005522	1449	RW,I,CON,LCL
$PDATA	000054	22	RW,D,CON,LCL
$IDATA	000216	71	RW,D,CON,LCL
$VARS	000400	128	RW,D,CON,LCL
$TEMPS	000034	14	RW,D,CON,LCL
LWING1	000170	60	RW,D,OVR,GBL

Total Space Allocated = 006640 1744

Total Virtual Array Storage = 78

```
0001          subroutine ltun(ip,inw,
                2       iga, iul, inc, iwng,
                2       la, theta, chi, c, b,
                2       xp, yp, zp, xc, yc, zc    )

0002          implicit character*200 (a-z)

0003          include 'lfly.inc'
      *   c       file [20,64]lfly.inc
0004  *           integer    pmax
0005  *           parameter (pmax = 90) ! max number of panels
0006  *           integer    nwmax
0007  *           parameter (nwmax=2)  ! max number of surfaces (wings)
0008  *           integer    nsmax
0009  *           parameter (nsmax=2)  ! max number of spanwise stations
0010  *           integer    ncmax
0011  *           parameter (ncmax=44)  ! max number of chordwise stations
      *   c                               on a wing including wake
0012  *           integer    itmax
0013  *           parameter (itmax=5) ! max number of total iterations
0014          include 'lwing1.com'
      *   c       file [20,64]lwing1.com
      *
0015  *           integer  nw,                ! number of surfaces
      *           1                           ! wing + wake(s), & tunnel
      *           2        ign,               ! number of gammas
      *           3        ipn                ! number of panels
0016  *           real     la1(nwmax),
      *           1        la2(nwmax),
      *           2        sw(nwmax),         ! half span = b/2
      *           2        cwak(nwmax),       ! length of wake
      *           3        cw(nwmax),         ! chord of wing
      *           4        x0(nwmax),
      *           5        z0(nwmax),
      *           6        alpha(nwmax),
      *           7        dihed(nwmax),
      *           8        eh,                ! tunnel ellipse height
      *           9        gbh,               ! tunnel ground board height
      *           1        reyn               ! reynolds number e-6
0017  *           integer  stype(nwmax)       ! surface type wing wake or tunnel
      *   c                                     >0 NACA 4 digit airfoil
      *   c                                     =0 NACA 64A005
      *   c                                     = -10 tunnel
      *   c
0018  *           integer  nc(nwmax),         ! wing chordwise panels
      *           2        nct(nwmax),        ! total chordwise panels
      *           2        ns(nwmax),         ! spanwise panels
      *           3       !bc(nwmax),         ! boundary condition
      *           4        nsep(nwmax,nsmax)! separation pt
      *   c                separates at nsep panels upstream of trailing edge
0019  *           integer  itws,              ! wake shape iterations
      *           6        itsp,              ! separation pt iterations
      *           7        itto               ! total iterations
      *
```

```
      *
0020  *         common /lwingl/ nw, ign, ipn, lal, la2, sw, cwak, cw, x0, z0,
      *         1               alpha, dihed, eh, gbh, reyn, stype,
      *         2               nc, nct, ns, nsep, itws, itsp, itto
      *
0021           include 'lwing2.var'
      *  c       file 'lwing2.var'
      *  c       changed from common block to allow use of virtual memory
      *
0022  *         integer iga,
      *         1       iul,
      *         2       inc,
      *         3       iwng
0023  *         real    la,
      *         1       theta,
      *         2       chi,
      *         3       c,
      *         4       b,
      *         5       xp, yp, zp,
      *         6       xc, yc, zc
      *
0024  *        virtual    iga(pmax), iul(pmax), inc(pmax), iwng(pmax),
      *       1           la(pmax), theta(pmax), chi(pmax), c(pmax), b(pmax),
      *       2           xp(pmax), yp(pmax), zp(pmax),
      *       3           xc(pmax), yc(pmax), zc(pmax)
      *

0025          real      x1(4),y1(4),z1(4)  ! wing ref frame
0026          real      x2(4),y2(4),z2(4)  ! global ref frame
0027          real      x3(4),y3(4),z3(4)  ! panel ref frame

0028          integer incl,       ! chordwise paneling counter
                1       ins,        ! spanwise paneling counter
                2       inw,        ! wing counter
                3       iunit,
                4       j,
                5       iull,       ! surface counter
                6       ipt,ip

0029          real      sth,cth,cch,sch
0030          real      xt,yt,zt
0031          real    xp3
0032          real    pi

0033          real    xple,     ! x fraction of chord for panel leading edge
              1       xpte      ! x fraction of chord for panel trailing edge

0034          real      xtp(10),ytp(10),ztp(10)
0035          real      xtc(10),ytc(10),ztc(10)
0036          real      stp(10)

0037          real      y, phi0, phi,z,zmax,ymax


0038          pi = acos(-1.0)
```

```
PDP-11 FORTRAN-77 V5.0-0        15:03:59    13-Nov-87          Page 17
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0039          iunit = 1

0040          zmax = sw(inw)
0041          ymax = eh/2.

        d       type 501, ' zmax,ymax ', zmax,ymax
        d501    format (' ',a,5f7.2)

        c       compute phi0

0042          y = -ymax + gbh*eh
0043          z = zmax * sqrt(1.0 - (y/ymax)**2)
0044          phi0 = atan2(z,y)

        d       type 502, '   y,z,phi0 ',y,z,phi0*d
        d502    format  (a,3f8.3)
        d       type *, ' '

        c       calculate the control and panel edge points on the ellipse

        d       type *,' '
        d       type *,'  phi, ycp, zcp, slope, chi , ins '
        d       type *,' '

0045          do 101 ins = 1,ns(inw)
0046              phi = phi0 - (2*ins-1) *  phi0/(2*ns(inw))
0047              ytc(ins)  = sign(1., cos(phi))
              %             * zmax /sqrt( (zmax/ymax)**2 + tan(phi)**2 )
0048              ztc(ins) = ytc(ins) * tan(phi)

0049              stp(ins) = sign(zmax/ymax*ytc(ins)/sqrt( ymax**2 - ytc(ins)**2 ),
              %                   -tan(phi) )

        cd        tchi(ins)  = atan2( sign(stp(ins),-cos(phi)  )
        cd      1                   ,sign(1.      , sin(phi)  )  )

        cd          TYPE 503, '  ',phi*D,ytc(ins),ztc(ins),stp(ins),
        cd      1      tCHI(ins),ins

        d503    format(a,5f8.4,i8)
0050    101   continue    !     end do

        d       type *, ' '
        d       type *, '    ins,  ytp(ins), ztp(ins)   '
        d       type *, ' '

0051          ytp(1)    = - ymax  + gbh*2*ymax
0052          ztp(1)    = ztc(1)  - (ytc(1)-ytp(1)) * Stp(1)

        d       type *,  1.0,  ytp(1), ztp(1)

0053          do 102 ins = 2,ns(inw)+1
0054              ytp(ins) = (ztc(ins)-stp(ins)*ytc(ins) -
              *             (ztc(ins-1)-stp(ins-1)*ytc(ins-1)))
              @             / (stp(ins-1) - stp(ins))
```

```
PDP-11 FORTRAN-77 V5.0-0        15:03:59    13-Nov-87          Page 18
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0055              ztp(ins) = stp(ins)*ytp(ins) + (ztc(ins)-stp(ins)*ytc(ins))

        d           type *, ins,ytp(ins),ztp(ins)
0056    102     continue

        d         type *, ' '
        d         type *,' wing ',inw

0057            do 103 iul1 = 1,2                       ! upper/lower surface
0058              do 112 ins = 1,ns(inw)                ! span divisions
0059                do 121 incl = 1,nc(inw)                 ! wing ref frame

0060            ip = ip + 1

0061            xple  = (incl-1.)/nc(inw) * cw(inw)
0062            xpte  = (incl*1.)/nc(inw) * cw(inw)

0063            xl(1) = xple
0064            yl(1) = ytp(ins)
0065            zl(1) = ztp(ins)
0066            if (iul1.eq.1)  zl(1) = -zl(1)

0067            xl(2) = xpte
0068            yl(2) = ytp(ins)
0069            zl(2) = ztp(ins)
0070            if (iul1.eq.1)  zl(2) = -zl(2)

0071            xl(3) = xple
0072            yl(3) = ytp(ins+1)
0073            zl(3) = ztp(ins+1)
0074            if (iul1.eq.1)  zl(3) = -zl(3)

0075            xl(4) = xpte
0076            yl(4) = ytp(ins+1)
0077            zl(4) = ztp(ins+1)
0078            if (iul1.eq.1)  zl(4) = -zl(4)

        cd          type *, ' ip, xl...     '
        cd        do 301 j=1,4
        cd          type 501, ip, xl(j), yl(j), zl(j),
        cd        1       xl(j)-xl(1), yl(j)-yl(1), zl(j)-zl(1)
        cd501       format(i8,10f8.4)
        cd301     continue
        cd          type *, '     '
        cd        type *,' alpha,dihed ',  alpha(inw), dihed(inw)

        c         find coordinates in global reference frame
        c         rotate for dihedral and alpha, then translate by x0,z0


        c         x,y,z => global frame

0079            x2(1) = x0(inw) +  xl(1)
0080            y2(1) =            yl(1)
0081            z2(1) = z0(inw) +  zl(1)
```

```
0082            x2(2) = x0(inw) +  x1(2)
0083            y2(2) =            y1(2)
0084            z2(2) = z0(inw) +  z1(2)

0085            x2(3) = x0(inw) +  x1(3)
0086            y2(3) =            y1(3)
0087            z2(3) = z0(inw) +  z1(3)

0088            x2(4) = x0(inw) +  x1(4)
0089            y2(4) =            y1(4)
0090            z2(4) = z0(inw) +  z1(4)

        cd        type *, '  ip x2...    '
        cd      do 302 j=1,4
        cd        type 501, ip, x2(j), y2(j), z2(j),
        cd      1       x2(j)-x2(1), y2(j)-y2(1), z2(j)-z2(1)
        cd302   continue
        cd        type *, '                '

        c       determine individual panel parameters

0091            if (ip.eq.1) then
0092              iga(ip) = 1
0093            else
0094              iga(ip) = iga(ip-1)+1
0095              if (incl.eq.1) iga(ip) = iga(ip) + 1
0096            end if

0097            iul(ip)   = iul1
0098            inc(ip)   = incl
0099            iwng(ip)  = inw
0100            theta(ip) = 0.0

0101            chi(ip) = atan2( (z2(3)-z2(1))*cos(theta(ip))-
                1                (x2(3)-x2(1))*sin(theta(ip)) ,
                1                y2(3)-y2(1)   )

        d       type *, ' theta(ip), chi(ip) ', theta(ip), chi(ip)

0102            sch = sin(theta(ip))
0103            cth = cos(theta(ip))
0104            sch = sin(chi(ip))
0105            cch = cos(chi(ip))

        c       type *, ' sth,cth,sch,cch ',sth,cth,sch,cch

0106            do 104 ipt = 1,4

        c         find corner points in panel system (x3,y3,z3)  (z3 == 0)
        c         translate axis and rotate into panel system

0107              xt = x2(ipt) - x2(1)
0108              yt = y2(ipt) - y2(1)
0109              zt = z2(ipt) - z2(1)
```

```
0110              x3(ipt) =   xt*cth                  + zt*sth
0111              y3(ipt) = - xt*sth*sch + yt*cch     + zt*cth*sch
0112              z3(ipt) = - xt*sth*cch - yt*sch     + zt*cth*cch

0113    104       continue


        cd        type *, '  ip x3...  '
        cd        do 304 j=1,4
        cd          type 501, ip, x3(j), y3(j), z3(j),
        cd        1       x3(j)-x3(1), y3(j)-y3(1), z3(j)-z3(1)
        c304      continue
        cd        type *, '                '

0114              b(ip) = y3(3)

        c         approximate the swept tapered panel by a swept panel

0115              la(ip) = ( x3(3)        /b(ip) +
                  1        (x3(4)-x3(2))/b(ip)   )/2.

0116              c(ip) = (x3(2) + (x3(4)-x3(3)) )/2.

        c         inboard leading edge point of approximate panel in
        c         tapered panel reference frame(3)
0117              xp3   = x3(3)/2. - la(ip)*b(ip)/2.
        d         type *, ' xp3 ',xp3
        c         yp3,zp3 = 0

        c         determine the new pt 1 of the panel in global coordinates
        c         rotate, translate to global coordinates

0118              xp(ip) =  x2(1) + xp3*cth     !- yp3*sch*sth - zp3*cch*sth
0119              yp(ip) =  y2(1)               !+ yp3*cch     - zp3*sch
0120              zp(ip) =  z2(1) + xp3*sth     !+ yp3*sch*cth + zp3*cch*cth

        c         find control point of panel

0121              xc(ip) = (x2(1) + x2(2) + x2(3) + x2(4) )/4.
0122              yc(ip) = ytc(ins)
0123              if (iull.eq.1) then
0124                zc(ip) = - ztc(ins) + z0(inw)
0125              else
0126                zc(ip) =   ztc(ins) + z0(inw)
0127              end if

        d         write(iunit,504)
        d         1
        d         2  ip, iga(ip), iul(ip), inc(ip), iwng(ip),
        d         3
        d         4  la(ip), theta(ip), chi(ip), c(ip), b(ip),
        d         5
        d         6  xp(ip), yp(ip), zp(ip), xc(ip),yc(ip),zc(ip)
0128    504       format(5i4,5f10.4,x,2(3f9.4,x) )
```

```
0129            write(2)
                1
                2  ip, iga(ip), iul(ip), inc(ip), iwng(ip),
                3
                4  la(ip), theta(ip), chi(ip), c(ip), b(ip),
                5
                6  xp(ip), yp(ip), zp(ip), xc(ip),yc(ip),zc(ip)

0130    121       continue
0131    112     continue
0132    103   continue
0133          end
```

PROGRAM SECTIONS

Name	Size		Attributes
$CODE1	004354	1142	RW,I,CON,LCL
$PDATA	000022	9	RW,D,CON,LCL
$IDATA	000242	81	RW,D,CON,LCL
$VARS	000770	252	RW,D,CON,LCL
$TEMPS	000034	14	RW,D,CON,LCL
LWING1	000170	60	RW,D,OVR,GBL

Total Space Allocated = 006054 1558

```
0001          real function lwing(x,iul,naca)

0002          implicit character*200 (a-z)

        c       input
0003          real      x       ! 0 <= x <= 1 fraction of wing chord
0004          integer iul       ! 1 = upper, 2 = lower surface
0005          integer naca      ! naca 4 digit airfoil XXXX

0006          integer   ix,
                1       im,
                2       ip,
                3       ithick

0007          real      m,
                1       p,
                2       thick,
                3       z


0008          real  xa(26),ya(26)   ! coordinates for 64A006 airfoil
        c       mulitiplied by 5/6 to get approximate   64A005 airfoil
0009          DATA    xa        /0, 0.5, 0.75, 1.25, 2.5,
                1                 5.0, 7.5, 10, 15, 20,
                2                 25, 30, 35, 40, 45,
                3                 50, 55, 60, 65, 70,
                4                 75, 80, 85, 90, 95,
                5                 100/

0010          DATA    ya        /0,     0.485, 0.585, 0.739, 1.016,
                1                 1.399, 1.684, 1.919, 2.283, 2.557,
                2                 2.757, 2.896, 2.977, 2.999, 2.945,
                3                 2.825, 2.653, 2.438, 2.188, 1.907,
                4                 1.602, 1.285, 0.967, 0.649, 0.331,
                5                 0.013/

        d       type *,' lwing  x, iul , naca', x, iul , naca

0011          if (naca.le.0) then ! use 64A005 airfoil

0012            ix = 1
0013    201       ix = ix + 1
0014                if ((x*100.0).gt.xa(ix).and.ix.lt.26) goto 201

0015          z = ya(ix-1) +
              1  (x*100.-xa(ix-1))/(xa(ix)-xa(ix-1))*(ya(ix)-ya(ix-1))

0016            if (iul.eq.1) then ! upper surface
        d       type *, ' lwing ', z
0017              lwing =  Z/100   * 5./6.
0018            else
        d       type *, ' lwing ', -z
0019              lwing = -Z/100  * 5./6.
0020            end if
```

```
PDP-11 FORTRAN-77 V5.0-0        15:04:18    13-Nov-87          Page 23
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR


0021          else ! use naca airfoil

0022            im =     naca/1000.

0023            ip =     (naca-1000*im)/100

0024            ithick = (naca-1000*im-100*ip)

0025            m = im/100.
0026            p = ip/10.
0027            thick = ithick/100.

        d       type *, ' m,p,thick ', m,p,thick

        c       separate naca input into thick = thickness  0.01 to .99
        c       and     p = point of maximum chamber
        c       and     m = amount of maximum chamber
        c

0028            z  = 0.2969*x**.5 - 0.126*x - 0.3516*x**2 +
                1    0.2843*x**3 - 0.1015*x**4
        d       type *, ' x,z ', x,z

0029            if (iul.eq.1) then
0030                z   = z*thick*5.0
0031            else
0032                z   = - z*thick*5.0
0033            end if

        d         type *,' x,z,p ', x, z, p

0034            if ( x.lt.p) then
0035                lwing = z + m/p**2*(2*p*x-x**2)
        d           type *,' lwing ', z + m/p**2*(2*p*x-x**2)
0036            else
        d           type *,' lwing ', z + m/(1.-p)**2*((1.-2*p)+2*p*x-x**2)
0037                lwing = z + m/(1.-p)**2*((1.-2*p)+2*p*x-x**2)
0038            end if
0039          end if
0040          end


PROGRAM SECTIONS

 Name        Size                 Attributes

$CODE1   001002   257             RW,I,CON,LCL
$PDATA   000024    10             RW,D,CON,LCL
$IDATA   000004     2             RW,D,CON,LCL
$VARS    000350   116             RW,D,CON,LCL
$TEMPS   000004     2             RW,D,CON,LCL


Total Space Allocated = 001406   387
```

```
0001          subroutine lfly2

              c       input: reads the panel configuration file 'lfly2u.xxx'
              c       output: velocity coefficients in file 'lfly3u.xxx'

              c       At pressent no check is made to see if an influence
              c       coeffient is needed.  No calculation is needed if the
              c       panel and control point have not changed position
              c       thus wing and panel influenct on wing and  panel remain
              c       unchanged.


0002          implicit character*200 (a-z)

              c       all common blocks MUST occur in main program section
0003          include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0004  *         integer    pmax
0005  *         parameter (pmax = 90) ! max number of panels
0006  *         integer    nwmax
0007  *         parameter (nwmax=2)  ! max number of surfaces (wings)
0008  *         integer    nsmax
0009  *         parameter (nsmax=2)  ! max number of spanwise stations
0010  *         integer    ncmax
0011  *         parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0012  *         integer    itmax
0013  *         parameter (itmax=5) ! max number of total iterations
0014          include 'lchar.com'
     *  c       file [20,64]lchar.com
     *
0015  *         character*40  title
0016  *         character*4  run
     *
0017  *         common /lchar/  title,run
     *
0018          include 'lwingl.com'
     *  c       file [20,64]lwingl.com
     *
0019  *         integer  nw,                ! number of surfaces
     *         1                           ! wing + wake(s), & tunnel
     *         2        ign,               ! number of gammas
     *         3        ipn                ! number of panels
0020  *         real     la1(nwmax),
     *         1        la2(nwmax),
     *         2        sw(nwmax),         ! half span = b/2
     *         2        cwak(nwmax),       ! length of wake
     *         3        cw(nwmax),         ! chord of wing
     *         4        x0(nwmax),
     *         5        z0(nwmax),
     *         6        alpha(nwmax),
     *         7        dihed(nwmax),
     *         8        eh,                ! tunnel ellipse height
     *         9        gbh,               ! tunnel ground board height
```

```
PDP-11 FORTRAN-77 V5.0-0        15:04:22    13-Nov-87          Page 25
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

     *          1        reyn              ! reynolds number e-6
0021 *          integer  stype(nwmax)      ! surface type wing wake or tunnel
     *  c                                    >0 NACA 4 digit airfoil
     *  c                                    =0 NACA 64A005
     *  c                                    = -10 tunnel
     *  c
0022 *          integer  nc(nwmax),        ! wing chordwise panels
     *          2        nct(nwmax),       ! total chordwise panels
     *          2        ns(nwmax),        ! spanwise panels
     *          3       !bc(nwmax),        ! boundary condition
     *          4        nsep(nwmax,nsmax)! separation pt
     *  c                separates at nsep panels upstream of trailing edge
0023 *          integer  itws,             ! wake shape iterations
     *          6        itsp,             ! separation pt iterations
     *          7        itto              ! total iterations
     *
     *
0024 *          common /lwingl/ nw, ign, ipn, lal, la2, sw, cwak, cw, x0, z0,
     *          1               alpha, dihed, eh, gbh, reyn, stype,
     *          2               nc, nct, ns, nsep, itws, itsp, itto
     *
0025           include 'lwing2.var'
     *  c       file 'lwing2.var'
     *  c       changed from common block to allow use of virtual memory
     *
0026 *          integer iga,
     *          1       iul,
     *          2       inc,
     *          3       iwng
0027 *          real    la,
     *          1       theta,
     *          2       chi,
     *          3       c,
     *          4       b,
     *          5       xp, yp, zp,
     *          6       xc, yc, zc
     *
0028 *         virtual    iga(pmax), iul(pmax), inc(pmax), iwng(pmax),
     *        1           la(pmax), theta(pmax), chi(pmax), c(pmax), b(pmax),
     *        2           xp(pmax), yp(pmax), zp(pmax),
     *        3           xc(pmax), yc(pmax), zc(pmax)
     *

0029           real    qu, qv, qw
0030           virtual qu(pmax,pmax), qv(pmax,pmax), qw(pmax,pmax)

0031           integer ic,ig

0032           call l2inpu (iga, iul, inc, iwng,
                 2            la, theta, chi, c, b,
                 2            xp, yp, zp, xc, yc, zc    )

       d         type *, ' call lwngco'

0033           call lwngco (qu, qv, qw,
```

```
PDP-11 FORTRAN-77 V5.0-0        15:04:22    13-Nov-87          Page 26
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

                2           iga, iul, inc, iwng,
                2           la, theta, chi, c, b,
                2           xp, yp, zp, xc, yc, zc     )


0034          fname(1:7) = 'lfly2u.'    ! u - unformatted
0035          fname(8:10) = run(1:3)

        d       type*, ' opening file ', fname

0036          open(unit=1, name=fname, form='unformatted', status='new',
                1       dispose='keep')

        d       fname(1:7) = 'lfly2f.'   ! f - formatted
        d       type*, ' opening file ', fname

        d       open(unit=2, name=fname, status='new', dispose='keep')

        c       last rows of array are initialized to zero
0037            do 101 ic = 1,ign                          ! control pt ic
0038              do 111 ig = 1,ign                        ! gamma ig
0039                write(1)     ic, ig, qu(ic,ig),qv(ic,ig),qw(ic,ig)
        d           write(2,501) ic, ig, qu(ic,ig),qv(ic,ig),qw(ic,ig)
       d501         format(2i5,3f10.5)
0040    111       continue
0041    101     continue

0042            close(1)
        d       close(2)

0043            end


PROGRAM SECTIONS

 Name       Size                  Attributes

$CODE1  000412   133            RW,I,CON,LCL
$PDATA  000162    57            RW,D,CON,LCL
$IDATA  000220    72            RW,D,CON,LCL
$VARS   000314   102            RW,D,CON,LCL
LCHAR   000054    22            RW,D,OVR,GBL
LWING1  000170    60            RW,D,OVR,GBL


Total Space Allocated = 001574   446

Total Virtual Array Storage = 1599

No FPP Instructions Generated
```

```
PDP-11 FORTRAN-77 V5.0-0        15:04:30    13-Nov-87          Page 27
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0001          subroutine l2inpu ( iga, iul, inc, iwng,
             2                    la, theta, chi, c, b,
             2                    xp, yp, zp, xc, yc, zc    )

0002             implicit character*200 (a-z)

0003             include 'lfly.inc'
     *  c        file [20,64]lfly.inc
0004 *           integer    pmax
0005 *           parameter (pmax = 90) ! max number of panels
0006 *           integer    nwmax
0007 *           parameter (nwmax=2)  ! max number of surfaces (wings)
0008 *           integer    nsmax
0009 *           parameter (nsmax=2)  ! max number of spanwise stations
0010 *           integer    ncmax
0011 *           parameter (ncmax=44)  ! max number of chordwise stations
     *  c                                on a wing including wake
0012 *           integer    itmax
0013 *           parameter (itmax=5) ! max number of total iterations
0014             include 'lwingl.com'
     *  c        file [20,64]lwingl.com
     *
0015 *           integer  nw,                ! number of surfaces
     *           1                           ! wing + wake(s), & tunnel
     *           2        ign,               ! number of gammas
     *           3        ipn                ! number of panels
0016 *           real     la1(nwmax),
     *           1        la2(nwmax),
     *           2        sw(nwmax),         ! half span = b/2
     *           2        cwak(nwmax),       ! length of wake
     *           3        cw(nwmax),         ! chord of wing
     *           4        x0(nwmax),
     *           5        z0(nwmax),
     *           6        alpha(nwmax),
     *           7        dihed(nwmax),
     *           8        eh,                ! tunnel ellipse height
     *           9        gbh,               ! tunnel ground board height
     *           1        reyn               ! reynolds number e-6
0017 *           integer  stype(nwmax)       ! surface type wing wake or tunnel
     *  c                                     >0 NACA 4 digit airfoil
     *  c                                     =0 NACA 64A005
     *  c                                     = -10 tunnel
     *  c
0018 *           integer  nc(nwmax),         ! wing chordwise panels
     *           2        nct(nwmax),        ! total chordwise panels
     *           2        ns(nwmax),         ! spanwise panels
     *           3       !bc(nwmax),         ! boundary condition
     *           4        nsep(nwmax,nsmax)! separation pt
     *  c                 separates at nsep panels upstream of trailing edge
0019 *           integer  itws,              ! wake shape iterations
     *           6        itsp,              ! separation pt iterations
     *           7        itto               ! total iterations
     *
     *
0020 *           common /lwingl/ nw, ign, ipn, la1, la2, sw, cwak, cw, x0, z0,
```

```
PDP-11 FORTRAN-77 V5.0-0        15:04:30    13-Nov-87          Page 28
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

     *          1               alpha, dihed, eh, gbh, reyn, stype,
     *          2               nc, nct, ns, nsep, itws, itsp, itto
     *
0021             include 'lwing2.var'
     *  c       file 'lwing2.var'
     *  c       changed from common block to allow use of virtual memory
     *
0022 *           integer iga,
     *          1       iul,
     *          2       inc,
     *          3       iwng
0023 *           real    la,
     *          1       theta,
     *          2       chi,
     *          3       c,
     *          4       b,
     *          5       xp, yp, zp,
     *          6       xc, yc, zc
     *
0024 *        virtual    iga(pmax), iul(pmax), inc(pmax), iwng(pmax),
     *       1           la(pmax), theta(pmax), chi(pmax), c(pmax), b(pmax),
     *       2           xp(pmax), yp(pmax), zp(pmax),
     *       3           xc(pmax), yc(pmax), zc(pmax)
     *
0025             include 'lchar.com'
     *  c       file [20,64]lchar.com
     *
0026 *           character*40  title
0027 *           character*4  run
     *
0028 *           common /lchar/  title,run
     *

0029             character*10 fname

0030             integer ip
0031             real    pi
0032             integer iunit
0033             integer ipl
0034             integer inw

         d       type *,' l2inpu called '

0035             fname(1:7) = 'lflylu.'
0036             fname(8:10) = run(1:3)

         d       type *,' opening file ',fname

0037             open(unit=1,name=fname,form='unformatted',
                 1    status='old')

0038             pi = acos(-1.0)
0039             ip = 1                          !  panel counter
0040             iunit = 1
```

```
PDP-11 FORTRAN-77 V5.0-0        15:04:30    13-Nov-87          Page 29
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0041     201      continue
0042                read(1,err=901,end=902)
                 1
                 2  ipl, iga(ip), iul(ip), inc(ip), iwng(ip),
                 3
                 4  la(ip), theta(ip), chi(ip), c(ip), b(ip),
                 5
                 6  xp(ip), yp(ip), zp(ip), xc(ip), yc(ip), zc(ip)

         d502    format(5i4,12f6.2)
         d       type 502,
         d       1
         d       2  ip, iga(ip), iul(ip), inc(ip), iwng(ip),
         d       3
         d       4  la(ip), theta(ip), chi(ip), c(ip), b(ip),
         d       5
         d       6  xp(ip), yp(ip), zp(ip), xc(ip), yc(ip), zc(ip)


0043                ip = ip + 1
0044             goto 201

0045     901     continue
0046             type *,' error reading file ',fname

0047     902     continue
0048             close(1)

         d       if (ip-1.ne.ipn) then
         d         type *, ' input 2 ipn error, ip = ',ip
         d         call exit
         d       end if

         d       type *,' input 2 ipn = ',ipn

0049             end


PROGRAM SECTIONS

 Name        Size                 Attributes

$CODE1   001074   286             RW,I,CON,LCL
$PDATA   000104    34             RW,D,CON,LCL
$IDATA   000226    75             RW,D,CON,LCL
$VARS    000026    11             RW,D,CON,LCL
LWING1   000170    60             RW,D,OVR,GBL
LCHAR    000054    22             RW,D,OVR,GBL


Total Space Allocated = 001720   488

No FPP Instructions Generated
```

```
PDP-11 FORTRAN-77 V5.0-0        15:04:37    13-Nov-87          Page 30
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0001          subroutine lwnqco ( qu, qv, qw,
             2               iga, iul, inc, iwng,
             2               la, theta, chi, c, b,
             2               xp, yp, zp, xc, yc, zc )

        c       computes the velocity influence coeffients for each panel and control

0002          implicit character*200 (a-z)

0003          include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0004 *          integer    pmax
0005 *          parameter (pmax = 90) ! max number of panels
0006 *          integer    nwmax
0007 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0008 *          integer    nsmax
0009 *          parameter (nsmax=2)  ! max number of spanwise stations
0010 *          integer    ncmax
0011 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0012 *          integer    itmax
0013 *          parameter (itmax=5) ! max number of total iterations
0014          include 'lchar.com'
     *  c       file [20,64]lchar.com
     *
0015 *          character*40  title
0016 *          character*4  run
     *
0017 *          common /lchar/  title,run
     *
0018          include 'lwingl.com'
     *  c       file [20,64]lwingl.com
     *
0019 *          integer  nw,                ! number of surfaces
     *          1                           ! wing + wake(s), & tunnel
     *          2        ign,               ! number of gammas
     *          3        ipn                ! number of panels
0020 *          real     la1(nwmax),
     *          1        la2(nwmax),
     *          2        sw(nwmax),         ! half span = b/2
     *          2        cwak(nwmax),       ! length of wake
     *          3        cw(nwmax),         ! chord of wing
     *          4        x0(nwmax),
     *          5        z0(nwmax),
     *          6        alpha(nwmax),
     *          7        dihed(nwmax),
     *          8        eh,                ! tunnel ellipse height
     *          9        gbh,               ! tunnel ground board height
     *          1        reyn               ! reynolds number e-6
0021 *          integer  stype(nwmax)       ! surface type wing wake or tunnel
     *  c                                     >0 NACA 4 digit airfoil
     *  c                                     =0 NACA 64A005
     *  c                                     = -10 tunnel
     *  c
0022 *          integer  nc(nwmax),         ! wing chordwise panels
```

```
PDP-11 FORTRAN-77 V5.0-0        15:04:37    13-Nov-87          Page 31
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

     *          2        nct(nwmax),       ! total chordwise panels
     *          2        ns(nwmax),        ! spanwise panels
     *          3       !bc(nwmax),        ! boundary condition
     *          4        nsep(nwmax,nsmax)! separation pt
     *  c                separates at nsep panels upstream of trailing edge
0023 *          integer  itws,             ! wake shape iterations
     *          6        itsp,             ! separation pt iterations
     *          7        itto              ! total iterations
     *
     *
0024 *          common /lwingl/ nw, ign, ipn, lal, la2, sw, cwak, cw, x0, z0,
     *          1               alpha, dihed, eh, gbh, reyn, stype,
     *          2               nc, nct, ns, nsep, itws, itsp, itto
     *
0025           include 'lwing2.var'
     *  c       file 'lwing2.var'
     *  c       changed from common block to allow use of virtual memory
     *
0026 *          integer iga,
     *          1       iul,
     *          2       inc,
     *          3       iwng
0027 *          real    la,
     *          1       theta,
     *          2       chi,
     *          3       c,
     *          4       b,
     *          5       xp, yp, zp,
     *          6       xc, yc, zc
     *
0028 *         virtual    iga(pmax), iul(pmax), inc(pmax), iwng(pmax),
     *        1           la(pmax), theta(pmax), chi(pmax), c(pmax), b(pmax),
     *        2           xp(pmax), yp(pmax), zp(pmax),
     *        3           xc(pmax), yc(pmax), zc(pmax)
     *

0029           integer   ip
0030           real      qu, qv, qw
0031           virtual   qu(pmax,pmax), qv(pmax,pmax), qw(pmax,pmax)

0032           character*10 fname

0033           integer   i,j,ic,incl

0034           real      xl,yl,zl,bl
0035           real      x2,y2,z2,b2

0036           real      sth,cth,sch,cch

0037           real      ubl,vbl,wbl
0038           real      ual,val,wal

0039           real      ub2,vb2,wb2
0040           real      ua2,va2,wa2
```

```
0041          real      vs1,ws1
0042          real      vs2,ws2

0043          real      ua1g,va1g,wa1g
0044          real      ua2g,va2g,wa2g

0045          real      ub1g,vb1g,wb1g
0046          real      ub2g,vb2g,wb2g

0047          real      us1g,vs1g,ws1g
0048          real      us2g,vs2g,ws2g

0049          real      uag,vag,wag
0050          real      ubg,vbg,wbg
0051          real      usg,vsg,wsg

0052          real    vinf,winf

0053          integer neq,ig,inw


0054          do 101 i = 1,ign    ! clear out the needed part of matrix
0055            do 111 j = 1,ign
0056              qu(i,j)=0
0057              qv(i,j)=0
0058              qw(i,j)=0
0059    111     continue
0060    101   continue

        d     type *,' arrays clear  starting , ign  ',ign

        C----------- PRIMED coordinates for control pt

0061          DO  102 ip = 1,ipn  ! each panel         -> ip

        d       type*, ' ip, xp(ip),yp(ip),zp(ip) ',ip, xp(ip),yp(ip),zp(ip)

0062            sth = sin(theta(ip))
0063            cth = cos(theta(ip))
0064            sch = sin(chi(ip))
0065            cch = cos(chi(ip))

        d       type *, 'ip, theta,chi ', ip, theta(ip) ,chi(ip)

0066            do 112 ic = 1,ipn   ! at each conrol pt  -> ic
        c         coordinates of control point in prime system (panel chord'=1)


        d           type *, '  ip , ic = ',ip,ic

        d           type *, ' ic,xc(ic),yc(ic),zc(ic) ', ic,xc(ic),yc(ic),zc(ic)

        c           xl =   x*cth                  + z*sth
        c           yl = - x*sth*sch + y*cch      + z*cth*sch
```

```
        c           zl = - x*sth*cch - y*sch      + z*cth*cch

0067                xl   =( ( xc(ic) - xp(ip) )*cth
               1              +( zc(ic) - zp(ip) )*sth       )/c(ip)

0068                yl   =(-( xc(ic) - xp(ip) )*sth*sch
               1              +( yc(ic) - yp(ip) )    *cch
               1              +( zc(ic) - zp(ip) )*cth*sch   )/c(ip)

0069                zl   =(-( xc(ic) - xp(ip) )*sth*cch
               1              -( yc(ic) - yp(ip) )    *sch
               1              +( zc(ic) - zp(ip) )*cth*cch   )/c(ip)

0070                bl   = b(ip) /c(ip)

        d       type *,  '  xl,yl,zl,bl  ' , xl,yl,zl,bl

        C                    IMAGE
        c Image system of panels  x,-y,z

0071                x2   =( ( xc(ic) - xp(ip) )*cth
               1              +( zc(ic) - zp(ip) )*sth       )/c(ip)

0072                y2   =(-( xc(ic) - xp(ip) )*sth*sch
               1              +(-yc(ic) - yp(ip) )    *cch
               1              +( zc(ic) - zp(ip) )*cth*sch   )/c(ip)

0073                z2   =(-( xc(ic) - xp(ip) )*sth*cch
               1              -(-yc(ic) - yp(ip) )    *sch
               1              +( zc(ic) - zp(ip) )*cth*cch   )/c(ip)

0074                b2   = b(ip)/c(ip)

        d           type *,  '  x2,y2,z2,b2  ' , x2,y2,z2,b2


        c       to find the effect of the image panels the effect of the original
        c       panel on the image of the control point is calculated the relations
        c       from symmetry are u = u' w = w' v = -v' where u,v,w are the
        c       effects of the image panel on the control point and u',v',w'
        c       are the effects of the panel on the image control point - global fra

        c  the influence is spilt into the following parts
        c       1) the panel itself (u,w) local/ (u,w)global
        c       2) the bound vorticity at the panel edges from upstream panels ""
        c       3) the trailing vorticity to infinity (u,v,w) 1/(v,w)g.

        c           find the influence due to the panel itself

        c           special case of control pt on panel
0075                if (ic.eq.ip.and.iul(ip).eq.1) then ! top panel
0076                  call lpanel(xl,   yl, zl,bl, la(ip), 1.,ubl,vbl,wbl)  ! gamma b
0077                  call lpanel(1.-xl,yl,-zl,bl,-la(ip),-1.,ual,val,wal)  ! gamma a
0078                else if (ic.eq.ip.and.iul(ip).eq.2) then ! bottom panel
0079                  call lpanel(xl,   yl, zl,bl, la(ip),-1.,ubl,vbl,wbl)  ! gamma b
```

```
0080                call lpanel(1.-xl,yl,-zl,bl,-la(ip), 1.,ual,val,wal)  ! gamma a
0081               else  ! control pt not on surface
0082                call lpanel(xl,   yl, zl,bl, la(ip), 1.,ubl,vbl,wbl)  ! gamma b
0083                call lpanel(1.-xl,yl,-zl,bl,-la(ip), 1.,ual,val,wal)  ! gamma a
0084               end if

0085               ual = -ual
0086               wal = -wal

       c           influence of panel on image control pt
0087               call lpanel(x2,   y2, z2,b2, la(ip), 1.,ub2,vb2,wb2)
0088               call lpanel(1.-x2,y2,-z2,b2,-la(ip), 1.,ua2,va2,wa2)
0089               ua2 = -ua2
0090               wa2 = -wa2

       c           effect of the constant strength vortex along sides of panel
0091               call lside(xl,yl,zl,bl,lal,vsl,wsl)
0092               val = val + 0.5*vsl
0093               wal = wal + 0.5*wsl

       c           effect of the constant strength vortex along sides of panel
0094               call lside(x2,y2,z2,b2,la2,vs2,ws2)
0095               va2 = va2 + 0.5*vs2
0096               wa2 = wa2 + 0.5*ws2

       c       convert to global coordinates ie veloity/unit capital gamma
       c       where capital gamma is in global coordinates

0097               vsl = vsl/c(ip)
0098               wsl = wsl/c(ip)
0099               vs2 = vs2/c(ip)
0100               ws2 = ws2/c(ip)

       c           vsl = 0
       c           wsl = 0
       c           vs2 = 0
       c           ws2 = 0

       c           the chord length is factored out in prime coordinates
       c           coorect to global by /chord(i,j)

       c           velocitys in global frame

       c           x =   xl*cth     - yl*sch*sth - zl*cch*sth
       c           y =              yl*cch     - zl*sch
       c           z =   xl*sth     + yl*sch*cth + zl*cch*cth

0101               ualg =  ual*cth - val*sch*sth - wal*cch*sth
0102               valg =            val*cch     - wal*sch
0103               walg =  ual*sth + val*sch*cth + wal*cch*cth

0104               ua2g =  ua2*cth - va2*sch*sth - wa2*cch*sth
0105               va2g =            va2*cch     - wa2*sch
0106               wa2g =  ua2*sth + va2*sch*cth + wa2*cch*cth
```

```
0107                ublg =  ubl*cth - vbl*sch*sth - wbl*cch*sth
0108                vblg =            vbl*cch     - wbl*sch
0109                wblg =  ubl*sth + vbl*sch*cth + wbl*cch*cth

0110                ub2g =  ub2*cth - vb2*sch*sth - wb2*cch*sth
0111                vb2g =            vb2*cch     - wb2*sch
0112                wb2g =  ub2*sth + vb2*sch*cth + wb2*cch*cth

0113                uslg =          - vsl*sch*sth - wsl*cch*sth
0114                vslg =            vsl*cch     - wsl*sch
0115                wslg =            vsl*sch*cth + wsl*cch*cth

0116                us2g =          - vs2*sch*sth - ws2*cch*sth
0117                vs2g =            vs2*cch     - ws2*sch
0118                ws2g =            vs2*sch*cth + ws2*cch*cth


0119                uag = ualg + ua2g
0120                ubg = ublg + ub2g

0121                vag = valg - va2g
0122                vbg = vblg - vb2g

0123                wag = walg + wa2g
0124                wbg = wblg + wb2g

0125                usg = uslg + us2g
0126                vsg = vslg - vs2g
0127                wsg = wslg + ws2g


        c       use gamma a array to index influence

0128                qu(ic,iga(ip)) = qu(ic,iga(ip)) + uag   !
0129                qv(ic,iga(ip)) = qv(ic,iga(ip)) + vag   ! add "a"
0130                qw(ic,iga(ip)) = qw(ic,iga(ip)) + wag   ! contribution

0131                qu(ic,iga(ip)+1) = qu(ic,iga(ip)+1) + ubg   !
0132                qv(ic,iga(ip)+1) = qv(ic,iga(ip)+1) + vbg   ! add "b"
0133                qw(ic,iga(ip)+1) = qw(ic,iga(ip)+1) + wbg   ! contribution


        c       effect of the vorticity trailing off panels  on the sides of
        c       panel (i,j) on cp (ic,jc)

0134                if (inc(ip).gt.1) then          ! not first panel

0135                  qu(ic,iga(ip)-inc(ip)+1) = qu(ic,iga(ip)-inc(ip)+1)
                1     + 0.5*c(ip-inc(ip)+1)*usg
0136                  qv(ic,iga(ip)-inc(ip)+1) = qv(ic,iga(ip)-inc(ip)+1)
                1     + 0.5*c(ip-inc(ip)+1)*vsg
0137                  qw(ic,iga(ip)-inc(ip)+1) = qw(ic,iga(ip)-inc(ip)+1)
                1     + 0.5*c(ip-inc(ip)+1)*wsg

0138                  qu(ic,iga(ip)) = qu(ic,iga(ip))
```

```
PDP-11 FORTRAN-77 V5.0-0        15:04:37    13-Nov-87          Page 36
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

                1     + 0.5*c(ip-1)*usg
0139                  qv(ic,iga(ip)) = qv(ic,iga(ip))
                1     + 0.5*c(ip-1)*vsg
0140                  qw(ic,iga(ip)) = qw(ic,iga(ip))
                1     + 0.5*c(ip-1)*wsg

0141                  do 121 incl = 2,inc(ip)-1,1

0142                    qu(ic,iga(ip)-inc(ip)+incl) = qu(ic,iga(ip)-inc(ip)+incl)
                1       + 0.5*(c(ip-inc(ip)+incl)+c(ip-inc(ip)+incl-1))*usg

0143                    qv(ic,iga(ip)-inc(ip)+incl) = qv(ic,iga(ip)-inc(ip)+incl)
                1       + 0.5*(c(ip-inc(ip)+incl)+c(ip-inc(ip)+incl-1))*vsg

0144                    qw(ic,iga(ip)-inc(ip)+incl) = qw(ic,iga(ip)-inc(ip)+incl)
                1       + 0.5*(c(ip-inc(ip)+incl)+c(ip-inc(ip)+incl-1))*wsg

0145    121           continue ! end do incl
0146                 end if


0147                 if (inc(ip).eq.nc(iwng(ip))) then ! last chordwise panel
0148                   call lvinf(xp(ip),yp(ip),zp(ip),
                1                 theta(ip),chi(ip),c(ip),b(ip),
                2                 xc(ic),yc(ic),zc(ic),
                3                 vinf,winf                        )

0149                   do 122 incl = 1, inc(ip),1
0150                     qv(ic,iga(ip)-inc(ip)+incl) = qv(ic,iga(ip)-inc(ip)+incl)
                1          + 0.5* c(ip-nc(iwng(ip))+incl) * vinf
0151                     qw(ic,iga(ip)-inc(ip)+incl) = qw(ic,iga(ip)-inc(ip)+incl)
                1          + 0.5* c(ip-nc(iwng(ip))+incl) * winf

0152                     qv(ic,iga(ip)-inc(ip)+incl+1) = qv(ic,iga(ip)-inc(ip)+incl+1)
                1          + 0.5* c(ip-nc(iwng(ip))+incl) * vinf
0153                     qw(ic,iga(ip)-inc(ip)+incl+1) = qw(ic,iga(ip)-inc(ip)+incl+1)
                1          + 0.5* c(ip-nc(iwng(ip))+incl) * winf
0154    122            continue ! end do incl
0155                  end if
        c            end if
0156    112       continue
0157    102    continue
0158           end


PROGRAM SECTIONS

 Name        Size                 Attributes

$CODE1   007332  1901            RW,I,CON,LCL
$PDATA   000050    20            RW,D,CON,LCL
$IDATA   000506   163            RW,D,CON,LCL
$VARS    000376   127            RW,D,CON,LCL
$TEMPS   000040    16            RW,D,CON,LCL
LCHAR    000054    22            RW,D,OVR,GBL
```

```
LWING1  000170    60            RW,D,OVR,GBL


Total Space Allocated = 011012  2309
```

```
PDP-11 FORTRAN-77 V5.0-0        15:05:04    13-Nov-87          Page 38
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0001            subroutine lpanel(x,y,z,b,la,inv,u,v,w)
0002            implicit character*200 (a-z)

        c       input   x,y,z   location of control pt
        c               b       panel span
        c               la      tan(sweep angle)
        c               inv     -1 if control pt is on bottem of panel
        c       output  u,v,w   induced velocities from panel with constant
        c                       span and linear chord gamma distribution
        c                       (ga(LE)=0 ga(TE) = 1.0 )
        c                       The vortex lines bend at the edges and trail
        c                       to the back of the panel on both sides.
        c                       1/(4*pi) has been factered out

        c
        c       Induced velocities are figured in two steps.
        c       1) infinite sheet starting at the left edge of the panel
        c       2) infinite sheet starting at the right edge of the panel
        c       The vortex lines start at the left trailing edge run forward
        c       and turn parallel to the leading and trailing edges.


0003            real x,y,z,la,b,inv,  u,v,w, ul,vl,wl

        d       type 501, ' lpanel x,y,z,la,b ',x,y,z,la,b
0004    501     format(a,6f6.2)


        c       u = 0
        c       v = 0
        c       w = 0
        c       return

0005            call lpanl (x      ,y   ,z , la, inv, u ,v ,w )
0006            call lpanl (x-la*b ,y-b ,z , la, inv, ul,vl,wl)
0007            u = u - ul
0008            v = v - vl
0009            w = w - wl

0010            end


PROGRAM SECTIONS

 Name       Size                 Attributes

$CODE1  000274    94             RW,I,CON,LCL
$IDATA  000044    18             RW,D,CON,LCL
$VARS   000014     6             RW,D,CON,LCL


Total Space Allocated = 000354   118
```

```
PDP-11 FORTRAN-77 V5.0-0        15:05:06    13-Nov-87          Page 39
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR


0001            subroutine lpanl (x,y,z,la,inv,   u,v,w)
0002            implicit character*200 (a-z)


        c       input (x,y,z,la,inv)
        c       output (u,v,w)

0003            real x,y,z,la,inv, u,v,w

        c       input: relative control point location, panel sweep angel
        c       output: the induced velocity due to unit gamma at the trailing edge of
        c       the panel + the bound vorticty turned back on the inbourd edge of
        c       the panel.  Panel has infinite span
        c       NOTE:   1/4*PI HAS BEEN FACTORED OUT

        c       check for contribution from the bound edge vorticity 1/28/87
        c       need to check panel coef.  recorded 1/28/87


0004            real r2,r,e2,e,s,f1,g1,g2,easinh,t,vcap

        c       easinh = e**(arc tan hyperbolic( ))


0005            if (abs(z).le.1.e-5) z = 0

0006            r2  = y**2 + z**2
0007            r   = sqrt( r2 )
0008            e2  = la**2 + 1
0009            e   = sqrt( e2 )
0010            s   = x - la*y

0011            f1  =   atan2( z*sqrt((x-1)**2+r2), (la*r2-y*(x-1)) )
                1     - atan2( z*sqrt( x   **2+r2), (la*r2-y* x   ) )

0012            g1  =log(
                1  easinh( (la*(x-1)+y)/sqrt((x-la*y-1)**2+z**2*(1+la**2)) )
                1 /easinh( (la*(x  )+y)/sqrt((x-la*y  )**2+z**2*(1+la**2)) ))

0013            g2  = log(    easinh( (x-1)/r  )
                1           / easinh( (x  )/r  )  )

0014            t = ( s*(e*g1-la*g2) - z*e2*f1 + y*g2         +
                1     la*( sqrt((x-1)**2+r2)-sqrt((x)**2+r2) )     )


        c       velocity of bound side vortex  missing    w = vcap* rp/(4*pi)

0015            vcap  = 0.5* ( (X**2*(X-1)/r2 + (x+1))/sqrt(r2+(x-1)**2)
                1              -(x**3      /r2 +  x   )/sqrt(r2+ x   **2)  +  g2 )

        c       type *,' vcap '
        c       type *,vcap
```

```
PDP-11 FORTRAN-77 V5.0-0        15:05:06    13-Nov-87          Page 40
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

        c       set to zero to measure panel effect
        c       vcap = 0.0 !

0016            u  =  inv* (-( s*f1 + z*(e*g1 - la*g2) ))
0017            v  =        -la*u  - vcap*z
0018            w  =        -t     + vcap*y

0019            end


PROGRAM SECTIONS

 Name       Size                 Attributes

$CODE1  001426   395             RW,I,CON,LCL
$PDATA  000004     2             RW,D,CON,LCL
$IDATA  000024    10             RW,D,CON,LCL
$VARS   000050    20             RW,D,CON,LCL
$TEMPS  000024    10             RW,D,CON,LCL


Total Space Allocated = 001552   437
```

```
PDP-11 FORTRAN-77 V5.0-0        15:05:10    13-Nov-87          Page 41
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0001            real function easinh(x)
       c        exp(arc_hyperbolic_sin(x) )
0002            implicit character*200 (a-z)
0003            real x

0004            if (x.lt.-55.) then
0005              easinh = ( -1./(2.*x))
0006            else
0007              easinh = ( x + sqrt(x**2+1.))
0008            end if

0009            return
0010            end


PROGRAM SECTIONS

 Name       Size                  Attributes

$CODE1  000130    44              RW,I,CON,LCL
$IDATA  000004     2              RW,D,CON,LCL


Total Space Allocated = 000134    46
```

```
PDP-11 FORTRAN-77 V5.0-0        15:05:11    13-Nov-87          Page 42
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR


0001            subroutine lside(x,y,z,b,la,v,w)
0002            implicit character*200 (a-z)

        c       input   x,y,z   relative location of control pt
        c               b       panel span
        c               la      tan(sweep angle)
        c       output  u,v,w   induced velocities from constant strength
        c                       vortices on panel sides trailing from upstream
        c                       panels

0003            real x,y,z
0004            real b,la
0005            real v,w
0006            real vl,wl

        c       v = 0
        c       w = 0
        c       return

0007            call lside1(x     ,  y,z, v, w)
0008            call lside1(x-b*la,y-b,z,vl,wl)
0009            v = v - vl
0010            w = w - wl

0011            return
0012            end


PROGRAM SECTIONS

 Name       Size                Attributes

$CODE1  000222    73            RW,I,CON,LCL
$IDATA  000030    12            RW,D,CON,LCL
$VARS   000010     4            RW,D,CON,LCL


Total Space Allocated = 000262    89
```

```
PDP-11 FORTRAN-77 V5.0-0        15:05:13    13-Nov-87          Page 43
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR


0001            subroutine lside1(x,y,z,v,w)
0002            implicit character*200 (a-z)

        c       input   x,y,z   relative control point location
        c       output  v,w     induced velocity due to a unit vortex from the origin
        c                       to the point (1,0,0)

0003            real x,y,z,v,w
0004            real rp2,vcap

0005            rp2 = (y**2+z**2)

0006            vcap = - ( (x-1)/sqrt((x-1)**2+rp2) - x/sqrt(x**2+rp2) )

0007            v  =   z/rp2 * vcap
0008            w  = - y/rp2 * vcap

0009            return
0010            end


PROGRAM SECTIONS

 Name       Size                Attributes

$CODE1  000200    64            RW,I,CON,LCL
$VARS   000010     4            RW,D,CON,LCL
$TEMPS  000004     2            RW,D,CON,LCL


Total Space Allocated = 000214    70
```

```
0001            subroutine lvinf(xp,yp,zp,theta,chi,c,b,xc,yc,zc,v,w)
0002            implicit character*200 (a-z)

0003            real xp,yp,zp
0004            real theta,chi,c,b
0005            real xc,yc,zc
0006            real v ,w
0007            real vi,wi,vli,wli

0008            real xpt1,ypt1,zpt1
0009            real xpt2,ypt2,zpt2

0010            real sth,cth,sch,cch

0011            real vl,wl

        c       v = 0
        c       w = 0
        c       return

0012            sth = sin(theta)
0013            cth = cos(theta)
0014            sch = sin(chi)
0015            cch = cos(chi)

0016            xpt1 = xp + c*cth
0017            ypt1 = yp
0018            zpt1 = zp + c*sth

0019            xpt2 = xp + c*cth - b*sch*sth
0020            ypt2 = yp         + b*cch
0021            zpt2 = zp + c*sth + b*sch*cth

        c       effect of panel
0022            call lvinf1(xc-xpt1,yc-ypt1,zc-zpt1,v ,w )
0023            call lvinf1(xc-xpt2,yc-ypt2,zc-zpt2,vl,wl)
0024            v = v - vl
0025            w = w - wl

        c       effect on image control point
0026            call lvinf1(xc-xpt1,-yc-ypt1,zc-zpt1,vi ,wi )
0027            call lvinf1(xc-xpt2,-yc-ypt2,zc-zpt2,vli,wli)
0028            vi = vi - vli
0029            wi = wi - wli

0030            v = v - vi
0031            w = w + wi

0032            return
0033            end
```

```
PDP-11 FORTRAN-77 V5.0-0        15:05:14    13-Nov-87          Page 45
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

PROGRAM SECTIONS

 Name        Size                 Attributes

$CODE1   000744    242            RW,I,CON,LCL
$IDATA   000060     24            RW,D,CON,LCL
$VARS    000100     32            RW,D,CON,LCL


Total Space Allocated = 001124   298
```

```
PDP-11 FORTRAN-77 V5.0-0        15:05:17    13-Nov-87          Page 46
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR


0001            subroutine lvinfl(x,y,z,v,w)
0002            implicit character*200 (a-z)

        c       input   x,y,z   relative control point location
        c       output  v,w     induced velocity from a vortex along
        c                       the positive x axis in the negative direction

0003            real x,y,z,v,w
0004            real rp,r,vcap


        d       type *,' lvinfl,x,y,z, ',  x,y,z

0005            rp = sqrt(y**2 + z**2)

0006            r  = sqrt(rp**2 + x**2)

0007            vcap = -1/rp *( -1. - x/r)

0008            v =  z/rp * vcap
0009            w = -y/rp * vcap
0010            return
0011            end

PROGRAM SECTIONS

 Name       Size                Attributes

$CODE1  000166    59            RW,I,CON,LCL
$VARS   000014     6            RW,D,CON,LCL

Total Space Allocated = 000202    65
```

```
0001          subroutine lfly3(clip,cdip,finish,stime)

        c       input:  lfly.xxx        wing definition
        c               lflylu.xxx      panel definition
        c               lfly2u.xxx      velocity influence coefficient matrices

        c       output: finish is .false. then
        c               set up to have influence coefficient matrices computed
        c               for next iteration
        c               finish is .true. then
        c               ouput lfly3u.xxx has been printed, last iteration has
        c               been computed
        c               lflylu.xxx is rewritten for the new panel formation
        c               solutions vector gamma, and velocities
        c               cl(iteration) in file lclvit.xxx
        c               wing wake in files    lwake(ns).xxx
        c               cl(ns) in file        lclvns.xxx
        c               cp(x)  in file        lcpvx(ns).xxx


0002          implicit character*200 (a-z)

0003          include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0004 *          integer    pmax
0005 *          parameter (pmax = 90) ! max number of panels
0006 *          integer    nwmax
0007 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0008 *          integer    nsmax
0009 *          parameter (nsmax=2)  ! max number of spanwise stations
0010 *          integer    ncmax
0011 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0012 *          integer    itmax
0013 *          parameter (itmax=5) ! max number of total iterations
0014          include 'lchar.com'
     *  c       file [20,64]lchar.com
     *
0015 *          character*40  title
0016 *          character*4  run
     *
0017 *          common /lchar/  title,run
     *
0018          include 'lwing1.com'
     *  c       file [20,64]lwing1.com
     *
0019 *          integer  nw,                ! number of surfaces
     *          1                           ! wing + wake(s), & tunnel
     *          2        ign,               ! number of gammas
     *          3        ipn                ! number of panels
0020 *          real     la1(nwmax),
     *          1        la2(nwmax),
     *          2        sw(nwmax),         ! half span = b/2
```

```
     *          2         cwak(nwmax),      ! length of wake
     *          3         cw(nwmax),        ! chord of wing
     *          4         x0(nwmax),
     *          5         z0(nwmax),
     *          6         alpha(nwmax),
     *          7         dihed(nwmax),
     *          8         eh,               ! tunnel ellipse height
     *          9         gbh,              ! tunnel ground board height
     *          1         reyn              ! reynolds number e-6
0021 *          integer   stype(nwmax)      ! surface type wing wake or tunnel
     *  c                                     >0 NACA 4 digit airfoil
     *  c                                     =0 NACA 64A005
     *  c                                     = -10 tunnel
     *  c
0022 *          integer   nc(nwmax),        ! wing chordwise panels
     *          2         nct(nwmax),       ! total chordwise panels
     *          2         ns(nwmax),        ! spanwise panels
     *          3        !bc(nwmax),        ! boundary condition
     *          4         nsep(nwmax,nsmax)! separation pt
     *  c                 separates at nsep panels upstream of trailing edge
0023 *          integer   itws,             ! wake shape iterations
     *          6         itsp,             ! separation pt iterations
     *          7         itto              ! total iterations
     *
     *
0024 *          common /lwing1/ nw, ign, ipn, la1, la2, sw, cwak, cw, x0, z0,
     *          1               alpha, dihed, eh, gbh, reyn, stype,
     *          2               nc, nct, ns, nsep, itws, itsp, itto
     *
0025          include 'lwing2.var'
     *  c       file 'lwing2.var'
     *  c       changed from common block to allow use of virtual memory
     *
0026 *          integer iga,
     *          1       iul,
     *          2       inc,
     *          3       iwng
0027 *          real    la,
     *          1       theta,
     *          2       chi,
     *          3       c,
     *          4       b,
     *          5       xp, yp, zp,
     *          6       xc, yc, zc
     *
0028 *        virtual    iga(pmax), iul(pmax), inc(pmax), iwng(pmax),
     *       1           la(pmax), theta(pmax), chi(pmax), c(pmax), b(pmax),
     *       2           xp(pmax), yp(pmax), zp(pmax),
     *       3           xc(pmax), yc(pmax), zc(pmax)
     *

0029          character*8       stime   ! start time

0030          logical   finish,         ! tells lfly if finished
             1          llblt           ! true if lblt called
```

```
0031          real      clip(pmax),
             1          cdip(pmax)
0032          integer   nsepol(nwmax,nsmax)
0033          integer   inw, ins

0034          real     qu,qv,qw
0035          virtual  qu(pmax,pmax),
             1         qv(pmax,pmax),
             2         qw(pmax,pmax)

0036          integer   ipi, ip
0037          virtual   ipi(nwmax,2,nsmax,ncmax)

0038          real      gamma(pmax),
             1          u(pmax) , v(pmax) , w(pmax),
             2          ul(pmax), vl(pmax), wl(pmax)

      c         for solving twice the same configuration with
      c         differect boundary conditions
      c         real    ul2(pmax),vl2(pmax),wl2(pmax)  ! need for one method of solut

0039          call l2inpu ( iga, iul, inc, iwng,
             2              la, theta, chi, c, b,
             2              xp, yp, zp, xc, yc, zc    )

0040          call l3inpu ( qu, qv, qw, ipi,
             2              iga, iul, inc, iwng,
             2              la, theta, chi, c, b,
             2              xp, yp, zp, xc, yc, zc    )

0041          if (itto.eq.1) then
0042            do 101 ip = 1,ipn
0043          clip(ip) = c(ip)*b(ip)*cos(theta(ip))*cos(chi(ip))*(3.-2.*iul(ip))
0044          cdip(ip) = c(ip)*b(ip)*sin(theta(ip))*cos(chi(ip))*(3.-2.*iul(ip))
0045    101     continue
0046          end if

0047          write(3,*) ' itto ,itsp, itws = ', itto ,itsp, itws

      c         the new coefficients are computed.

0048          finish = .true.
0049          call lveloc ( qu, qv, qw, gamma,
             1              u, v, w, ul, vl, wl,
             2              iga, iul, inc, iwng,
             2              la, theta, chi, c, b,
             2              xp, yp, zp, xc, yc, zc,
             2              finish                    )
      c         finish = true if normal velocities < eps


0050          do 102 inw = 1,nw
0051            do 111 ins = 1, ns(inw)
0052              nsepol(inw,ins) = nsep(inw,ins)
```

```
PDP-11 FORTRAN-77 V5.0-0        15:05:19    13-Nov-87          Page 50
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0053     111     continue
0054     102   continue


0055           llblt = (finish.and.itto.le.itmax)
0056           if (llblt)
              1    call lblt ( ul, vl, wl, ipi,
                 2             iga, iul, inc, iwng,
                 2             la, theta, chi, c, b,
                 2             xp, yp, zp, xc, yc, zc,
                 2             finish                    )
        c          finish = false if new separation pt found

0057             call lglob (    gamma, ipi,
              2                u, v, w, ul, vl, wl,
              2                iga, iul, inc, iwng,
              2                la, theta, chi, c, b,
              2                xp, yp, zp, xc, yc, zc, clip, cdip,
              2               stime, finish, llblt  )


        c        write(3,*) ' lfly3  calling lngeom '

0058           if (.not.finish) then
0059             call lngeom ( u, v, w, ul, vl, wl,
                 2             iga, iul, inc, iwng,
                 2             la, theta, chi, c, b,
                 2             xp, yp, zp, xc, yc, zc, nsepol  )

        c        increment values for next iteration
0060             itto = itto + 1
0061             if (llblt)  then
0062               itws = 1
0063               itsp = itsp + 1
0064             else
0065               itws = itws + 1
0066             end if
0067           end if

0068           end


PROGRAM SECTIONS

 Name        Size                 Attributes

$CODE1   001132    301            RW,I,CON,LCL
$PDATA   000212     69            RW,D,CON,LCL
$IDATA   000506    163            RW,D,CON,LCL
$VARS    004750   1268            RW,D,CON,LCL
$TEMPS   000024     10            RW,D,CON,LCL
LCHAR    000054     22            RW,D,OVR,GBL
LWING1   000170     60            RW,D,OVR,GBL
```

Total Space Allocated = 007312 1893

Total Virtual Array Storage = 1610

```
PDP-11 FORTRAN-77 V5.0-0        15:05:28    13-Nov-87          Page 52
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0001          subroutine l3inpu ( qu,qv,qw, ipi,
             2              iga, iul, inc, iwng,
             2              la, theta, chi, c, b,
             2              xp, yp, zp, xc, yc, zc    )

0002          implicit character*200 (a-z)

0003          include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0004 *          integer    pmax
0005 *          parameter (pmax = 90) ! max number of panels
0006 *          integer    nwmax
0007 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0008 *          integer    nsmax
0009 *          parameter (nsmax=2)  ! max number of spanwise stations
0010 *          integer    ncmax
0011 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0012 *          integer    itmax
0013 *          parameter (itmax=5) ! max number of total iterations
0014          include 'lchar.com'
     *  c       file [20,64]lchar.com
     *
0015 *          character*40  title
0016 *          character*4  run
     *
0017 *          common /lchar/  title,run
     *
0018          include 'lwingl.com'
     *  c       file [20,64]lwingl.com
     *
0019 *          integer  nw,              ! number of surfaces
     *          1                         ! wing + wake(s), & tunnel
     *          2        ign,             ! number of gammas
     *          3        ipn              ! number of panels
0020 *          real     la1(nwmax),
     *          1        la2(nwmax),
     *          2        sw(nwmax),       ! half span = b/2
     *          2        cwak(nwmax),     ! length of wake
     *          3        cw(nwmax),       ! chord of wing
     *          4        x0(nwmax),
     *          5        z0(nwmax),
     *          6        alpha(nwmax),
     *          7        dihed(nwmax),
     *          8        eh,              ! tunnel ellipse height
     *          9        gbh,             ! tunnel ground board height
     *          1        reyn             ! reynolds number e-6
0021 *          integer  stype(nwmax)     ! surface type wing wake or tunnel
     *  c                                   >0 NACA 4 digit airfoil
     *  c                                   =0 NACA 64A005
     *  c                                   = -10 tunnel
     *  c
0022 *          integer  nc(nwmax),       ! wing chordwise panels
     *          2        nct(nwmax),      ! total chordwise panels
     *          2        ns(nwmax),       ! spanwise panels
```

```
PDP-11 FORTRAN-77 V5.0-0        15:05:28    13-Nov-87          Page 53
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

     *          3       !bc(nwmax),        ! boundary condition
     *          4        nsep(nwmax,nsmax)! separation pt
     *  c                separates at nsep panels upstream of trailing edge
0023 *          integer  itws,              ! wake shape iterations
     *          6        itsp,              ! separation pt iterations
     *          7        itto               ! total iterations
     *
     *
0024 *          common /lwing1/ nw, ign, ipn, la1, la2, sw, cwak, cw, x0, z0,
     *          1               alpha, dihed, eh, gbh, reyn, stype,
     *          2               nc, nct, ns, nsep, itws, itsp, itto
     *
0025           include 'lwing2.var'
     *  c       file 'lwing2.var'
     *  c       changed from common block to allow use of virtual memory
     *
0026 *          integer iga,
     *          1       iul,
     *          2       inc,
     *          3       iwng
0027 *          real    la,
     *          1       theta,
     *          2       chi,
     *          3       c,
     *          4       b,
     *          5       xp, yp, zp,
     *          6       xc, yc, zc
     *
0028 *         virtual    iga(pmax), iul(pmax), inc(pmax), iwng(pmax),
     *        1           la(pmax), theta(pmax), chi(pmax), c(pmax), b(pmax),
     *        2           xp(pmax), yp(pmax), zp(pmax),
     *        3           xc(pmax), yc(pmax), zc(pmax)
     *

0029           real       q1, q2, q3
0030           real       qu, qv, qw
0031           virtual    qu(pmax,pmax),
              1           qv(pmax,pmax),
              2           qw(pmax,pmax)

0032           integer    ic1, ig1, ic, ig, i,
              1           inw, iul1, ins, incl

0033           integer    ipi
0034           virtual    ipi(nwmax,2,nsmax,ncmax)

0035           character*10       fname

0036           close(1)

0037           fname(1:7) = 'lfly2u.'    ! u - unformatted
0038           fname(8:10) = run(1:3)

0039           type*, ' opening file ', fname
```

```
PDP-11 FORTRAN-77 V5.0-0        15:05:28    13-Nov-87          Page 54
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0040          open(unit=1,name=fname,form='unformatted',status='old',
             1  dispose='keep')

        d     write(3,*) ' ic,ig,u,v,w as read in 13inpu '

        c     read in the arrays of influence coeffients
        c     note the bottem of the arrays are zero since there
        c     are more gammas than control points
        c     not reading in the last of the arrays gives
        c     an overflow probem in finding velocities
        c     since the matrix multiply routine is set up
        c     for square arrays

0041          do 101 ic = 1,ign     ! control pt ic
0042            do 111 ig = 1,ign   ! gamma ig

        c         read(1,err=901)  icl, igl, qu(ic,ig), qv(ic,ig), qw(ic,ig)
0043              read(1)          icl, igl, qu(ic,ig), qv(ic,ig), qw(ic,ig)
        d         write(3,*)icl, igl, qu(ic,ig), qv(ic,ig), qw(ic,ig)

        d         if (icl.ne.ic.or.igl.ne.ig) then
        d           type *, ' q read problem, icl,igl ', icl, igl
        d           call exit
        d         end if

0044    111     continue
0045    101   continue
        d     write(3,*) ' end of u,v,w from 13inpu '
0046          close(1)

        c       set up index for panel numbers given position of panel
        c
0047          i = 0
0048          do 102 inw = 1, nw
0049            do 112 iull = 1,2
0050              do 121 ins = 1,ns(inw)
0051                do 131 incl = 1,nct(inw)
0052                  i = i + 1
0053                  ipi(inw,iull,ins,incl) = i
0054    131         continue
0055    121       continue
0056    112     continue
0057    102   continue

0058          return
       d901    continue
       d      type *,' ic,ig ,icl,igl ', ic,ig ,icl,igl
       d      call exit
0059          end


PROGRAM SECTIONS

 Name        Size                 Attributes
```

$CODE1	001164	314	RW,I,CON,LCL
$PDATA	000072	29	RW,D,CON,LCL
$IDATA	000276	95	RW,D,CON,LCL
$VARS	000050	20	RW,D,CON,LCL
$TEMPS	000010	4	RW,D,CON,LCL
LCHAR	000054	22	RW,D,OVR,GBL
LWING1	000170	60	RW,D,OVR,GBL

Total Space Allocated = 002100 544

No FPP Instructions Generated

```
PDP-11 FORTRAN-77 V5.0-0        15:05:38    13-Nov-87          Page 56
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0001          subroutine lmatso(n,a,b)

        c       input   n       number of equations
        c               av      matrix (virtual array)
        c               b       right hand side
        c
        c       output  b       solution vector
        C
        C  forward elimination and back substitution
        c       if there is room the virtual array can be
        c       copied to a normal array which speeds
        c       execution by a factor of 10x

0002          include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0003 *          integer    pmax
0004 *          parameter (pmax = 90) ! max number of panels
0005 *          integer    nwmax
0006 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0007 *          integer    nsmax
0008 *          parameter (nsmax=2)  ! max number of spanwise stations
0009 *          integer    ncmax
0010 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                                 on a wing including wake
0011 *          integer    itmax
0012 *          parameter (itmax=5) ! max number of total iterations

0013          integer n

        c       real*8  b(pmax),av
        c       real    bv(pmax),a(pmax,pmax)

0014          real      a,t,b(pmax)
0015          virtual a(pmax,pmax)

        c       do 101 i = 1,n
        c         do 111 j = 1,n
        c           a(i,j) = av(i,j)
        c111      continue
        cc        b(i) = bv(i)
        c101    continue

0016          do 102 idiag = 1,n

        c       a(idiag,idiag) = lower right of upper triangular sub-matrix

0017            amax = 0
0018            do 112 i = idiag,n
0019              if ( amax.lt.abs(a(i,idiag)) ) then
0020                amax = abs(a(i,idiag))
0021                imax = i
0022              end if
0023    112     continue
```

```
PDP-11 FORTRAN-77 V5.0-0        15:05:38    13-Nov-87          Page 57
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

        c       imax = row number of next piviot
        c       swap rows idiag, imax and normalize new idiag row
        c       note a(imax,idiag) is overwritten LAST

        c       order of swap works properly for swaping only one element with itself
0024            t = b(imax)/a(imax,idiag)
0025            b(imax) = b(idiag)
0026            b(idiag) = t

0027            if (imax.ne.idiag) type *,' swap ',idiag,imax

        c       order of swap works properly for swaping only one element with itself
0028            do 113 j = n,idiag,-1
0029              t           = a(imax,j)/a(imax,idiag)
0030              a(imax,j)   = a(idiag,j)
0031              a(idiag,j) = t
0032    113     continue

        c         elliminate the entries below a(idiag,idiag)

0033            do 114 i = idiag+1,n
0034              do 121 j = idiag+1,n
0035                a(i,j) = a(i,j) - a(i,idiag)*a(idiag,j)
0036    121       continue
0037                b(i)   = b(i)   - a(i,idiag)*b(idiag)
0038    114     continue

0039    102   continue

        c     end of elimination start back substitution

0040          do 103 i = n,1,-1
0041            do 115 j = i+1,n
0042              b(i) = b(i) - a(i,j)*b(j)
0043    115     continue
0044    103   continue

0045          end


PROGRAM SECTIONS

 Name       Size                  Attributes

$CODE1  001420   392              RW,I,CON,LCL
$PDATA  000006     3              RW,D,CON,LCL
$IDATA  000024    10              RW,D,CON,LCL
$VARS   000020     8              RW,D,CON,LCL
$TEMPS  000016     7              RW,D,CON,LCL


Total Space Allocated = 001510   420
```

```
PDP-11 FORTRAN-77 V5.0-0        15:05:43    13-Nov-87          Page 58
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0001          subroutine lmatmu(n,a,x,b)
        c       input   n,a,x
        c       output  b

0002          include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0003 *          integer    pmax
0004 *          parameter (pmax = 90) ! max number of panels
0005 *          integer    nwmax
0006 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0007 *          integer    nsmax
0008 *          parameter (nsmax=2)  ! max number of spanwise stations
0009 *          integer    ncmax
0010 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                                on a wing including wake
0011 *          integer    itmax
0012 *          parameter (itmax=5) ! max number of total iterations

0013          integer n ! size of array
0014          real      a,      ! nXn matrix
                1       x(pmax),! n vector
                2       b(pmax) ! right hand side
0015          virtual a(pmax,pmax)

0016          do 101 i = 1,n
0017            b(i) = 0
0018            do 111 j = 1,n
0019              b(i) = b(i) + a(i,j)*x(j)
0020              if (b(i).gt.1000) then
0021                type *, ' i,j,a(i,j),b(i),x(j) ',i,j,a(i,j),b(i),x(j)
0022              end if


0023    111     continue
0024    101   continue
0025          return
0026          end


PROGRAM SECTIONS

 Name        Size                 Attributes

$CODE1   000452   149            RW,I,CON,LCL
$PDATA   000026    11            RW,D,CON,LCL
$IDATA   000036    15            RW,D,CON,LCL
$VARS    000004     2            RW,D,CON,LCL
$TEMPS   000010     4            RW,D,CON,LCL


Total Space Allocated = 000552   181
```

```
PDP-11 FORTRAN-77 V5.0-0        15:05:46    13-Nov-87          Page 59
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR


0001          subroutine lveloc ( qu, qv, qw, gamma,
                1                   u, v, w, ul, vl, wl,
                2                   iga, iul, inc, iwng,
                3                   la, theta, chi, c, b,
                4                   xp, yp, zp, xc, yc, zc,
                5                   finish                    )

        c       input   the variables have been initialized and
        c               qu, qv, qw contain the velocity coefficients
        c
        c       output  the velocites have been computed and the convergence flag
        c               is set if the solution has converged.
        c
        c


0002          implicit character*200 (a-z)

0003          include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0004 *          integer    pmax
0005 *          parameter (pmax = 90) ! max number of panels
0006 *          integer    nwmax
0007 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0008 *          integer    nsmax
0009 *          parameter (nsmax=2)  ! max number of spanwise stations
0010 *          integer    ncmax
0011 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0012 *          integer    itmax
0013 *          parameter (itmax=5) ! max number of total iterations
0014          include 'lwingl.com'
     *  c       file [20,64]lwingl.com
     *
0015 *          integer  nw,                ! number of surfaces
     *          1                           ! wing + wake(s), & tunnel
     *          2        ign,               ! number of gammas
     *          3        ipn                ! number of panels
0016 *          real     la1(nwmax),
     *          1        la2(nwmax),
     *          2        sw(nwmax),         ! half span = b/2
     *          2        cwak(nwmax),       ! length of wake
     *          3        cw(nwmax),         ! chord of wing
     *          4        x0(nwmax),
     *          5        z0(nwmax),
     *          6        alpha(nwmax),
     *          7        dihed(nwmax),
     *          8        eh,                ! tunnel ellipse height
     *          9        gbh,               ! tunnel ground board height
     *          1        reyn               ! reynolds number e-6
0017 *          integer  stype(nwmax)       ! surface type wing wake or tunnel
     *  c                                     >0 NACA 4 digit airfoil
     *  c                                     =0 NACA 64A005
     *  c                                     = -10 tunnel
     *  c
```

```
0018 *          integer  nc(nwmax),        ! wing chordwise panels
     *          2        nct(nwmax),       ! total chordwise panels
     *          2        ns(nwmax),        ! spanwise panels
     *          3       !bc(nwmax),        ! boundary condition
     *          4        nsep(nwmax,nsmax)! separation pt
     *  c                separates at nsep panels upstream of trailing edge
0019 *          integer  itws,             ! wake shape iterations
     *          6        itsp,             ! separation pt iterations
     *          7        itto              ! total iterations
     *
     *
0020 *          common /lwing1/ nw, ign, ipn, la1, la2, sw, cwak, cw, x0, z0,
     *          1               alpha, dihed, eh, gbh, reyn, stype,
     *          2               nc, nct, ns, nsep, itws, itsp, itto
     *
0021          include 'lwing2.var'
     *  c       file 'lwing2.var'
     *  c       changed from common block to allow use of virtual memory
     *
0022 *          integer iga,
     *          1       iul,
     *          2       inc,
     *          3       iwng
0023 *          real    la,
     *          1       theta,
     *          2       chi,
     *          3       c,
     *          4       b,
     *          5       xp, yp, zp,
     *          6       xc, yc, zc
     *
0024 *        virtual    iga(pmax), iul(pmax), inc(pmax), iwng(pmax),
     *       1           la(pmax), theta(pmax), chi(pmax), c(pmax), b(pmax),
     *       2           xp(pmax), yp(pmax), zp(pmax),
     *       3           xc(pmax), yc(pmax), zc(pmax)
     *

0025          integer   ic,
             1          ig,
             2          neql,              ! equation number of b.c.
             3          inw,
             4          ins,
             5          i,
             6          j,
             7          iul1,
             8          incl

0026          character*10      fname
0027          logical           finish

0028          real      qu, qv, qw, qlw , qlu
0029          virtual qu(pmax,pmax),
                1       qv(pmax,pmax),
                2       qw(pmax,pmax),
                3       qlw(pmax,pmax)
```

```
0030          real      u(pmax),   v(pmax),   w(pmax),
                1       ul(pmax),  vl(pmax),  wl(pmax),
                2       gamma(pmax)

0031          real      xl,x2

        c       induce free stream velocities for different panel coordinate systems

0032          real pi

0033          real    sth,cth,sch,cch
0034          integer ip0,ip1,ip2,ip3,ip4,iunit, ip
0035          integer     ip5,ip6,ip7,ip8

0036          real eps

0037          pi = acos(-1.0)

        c       initialize to zero part of arrays reserved for boundary conditions
        c       2 equations per spanwise strip
        c       initialize right hand side to 0 for boundary conditions
        c       and -4pi V.n for the rest

0038          do 101 ic = ipn+1, ign
0039            do 111 ig = 1, ign
0040              qlw(ic,ig) = 0
0041    111     continue
0042            gamma(ic) =  0
0043    101   continue

0044          do 102 ic = 1,ipn
        c         type*, ' ic,theta,chi ', ic,theta(ic),chi(ic)

0045            sth = sin(theta(ic))
0046            cth = cos(theta(ic))
0047            sch = sin(chi(ic))
0048            cch = cos(chi(ic))

0049            do 112 ig = 1,ign
        c       qlu(ic,ig) =  qu(ic,ig)*cth                       + qw(ic,ig)*sth
        c       qlv(ic,ig) = -qu(ic,ig)*sth*sch + qv(ic,ig)*cch + qw(ic,ig)*cth*sch
0050            qlw(ic,ig) = -qu(ic,ig)*sth*cch - qv(ic,ig)*sch + qw(ic,ig)*cth*cch
0051    112     continue

        c         right hand side for qlw induce = - vinf dot normal
        c         the matix routine returns the solution vector in the
        c         place of the rhs

0052            gamma(ic) =  sth*cch*(4*pi)
0053    102   continue

        c       finish linear system by adding boundary conditions
0054          call lbc( qlw, qu, qw, gamma,
```

```
     1       iga, iul, inc, iwng,
     2       la, theta, chi, c, b,
     3       xp, yp, zp, xc, yc, zc    )


c      type *, ' call lmatso '
0055       call lmatso(ign,qlw,gamma) ! lmatso destroyes input matrix and rhs
c      type *, ' return from lmatso '


c      to local from global
c      xl =   x*cth                    + z*sth
c      yl = - x*sth*sch + y*cch        + z*cth*sch
c      zl = - x*sth*cch - y*sch        + z*cth*cch
c
c      to global from local
c      x =   xl*cth     - yl*sch*sth - zl*cch*sth
c      y =                yl*cch     - zl*sch
c      z =   xl*sth     + yl*sch*cth + zl*cch*cth

0056       call lmatmu(ign, qu, gamma, u)
0057       call lmatmu(ign, qv, gamma, v)
0058       call lmatmu(ign, qw, gamma, w)


c      add v infinity to panel velocity contribution
0059       do 104 ip = 1,ipn

0060         sth = sin(theta(ip))
0061         cth = cos(theta(ip))
0062         sch = sin(chi(ip))
0063         cch = cos(chi(ip))

0064         u(ip) = u(ip)/(4*pi)  + 1.0
0065         v(ip) = v(ip)/(4*pi)
0066         w(ip) = w(ip)/(4*pi)

0067         ul(ip)  = u(ip)*cth                  + w(ip)*sth
0068         vl(ip)  =-u(ip)*sth*sch + v(ip)*cch + w(ip)*cth*sch
0069         wl(ip)  =-u(ip)*sth*cch - v(ip)*sch + w(ip)*cth*cch

0070   104   continue

0071       ip = 0
0072       eps = 0.01

c      at the last control pt the normal velocity is not controlled
0073       do 105 inw = 1,nw
0074         do 113 iul1 = 1,2                    ! upper/lower surface
0075           do 121 ins = 1,ns(inw)             ! span divisions
0076             do 131 incl = 1,nct(inw)
0077               ip = ip + 1

0078         finish = (finish
          1         .and.((abs(wl(ip)).le.eps)
```

```
PDP-11 FORTRAN-77 V5.0-0         15:05:46    13-Nov-87           Page 63
LFLY.TMP;1                /F77/OP/TR:BLOCKS/WR

             1        .or. (inc(ip).eq.nc(iwng(ip)).and.nsep(inw,ins).eq.0)))
       c                    3   4  4       4    5  54          4        4     321
       c                      23   4  5  54        3
0079     131           continue
0080     121         continue
0081     113       continue
0082     105     continue

0083             end


PROGRAM SECTIONS

 Name        Size                  Attributes

$CODE1   002642    721             RW,I,CON,LCL
$PDATA   000014      6             RW,D,CON,LCL
$IDATA   000474    158             RW,D,CON,LCL
$VARS    000126     43             RW,D,CON,LCL
$TEMPS   000016      7             RW,D,CON,LCL
LWING1   000170     60             RW,D,OVR,GBL


Total Space Allocated = 003706   995

Total Virtual Array Storage = 507
```

```
0001          subroutine lglob ( gamma, ipi,
             1              u, v, w, ul, vl, wl,
             2              iga, iul, inc, iwng,
             2              la, theta, chi, c, b,
             2              xp, yp, zp, xc, yc, zc, clip, cdip,
             3              stime, finish, llblt )

        c       input the solution has been found
        c
        c       output the file lfly3f.xxx has been written
        c

0002          implicit character*200 (a-z)

0003          include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0004 *          integer    pmax
0005 *          parameter (pmax = 90) ! max number of panels
0006 *          integer    nwmax
0007 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0008 *          integer    nsmax
0009 *          parameter (nsmax=2)  ! max number of spanwise stations
0010 *          integer    ncmax
0011 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0012 *          integer    itmax
0013 *          parameter (itmax=5) ! max number of total iterations
0014          include 'lchar.com'
     *  c       file [20,64]lchar.com
     *
0015 *          character*40  title
0016 *          character*4  run
     *
0017 *          common /lchar/  title,run
     *
0018          include 'lwingl.com'
     *  c       file [20,64]lwingl.com
     *
0019 *          integer  nw,                ! number of surfaces
     *          1                           ! wing + wake(s), & tunnel
     *          2        ign,               ! number of gammas
     *          3        ipn                ! number of panels
0020 *          real     la1(nwmax),
     *          1        la2(nwmax),
     *          2        sw(nwmax),         ! half span = b/2
     *          2        cwak(nwmax),       ! length of wake
     *          3        cw(nwmax),         ! chord of wing
     *          4        x0(nwmax),
     *          5        z0(nwmax),
     *          6        alpha(nwmax),
     *          7        dihed(nwmax),
     *          8        eh,                ! tunnel ellipse height
     *          9        gbh,               ! tunnel ground board height
     *          1        reyn               ! reynolds number e-6
```

```
PDP-11 FORTRAN-77 V5.0-0        15:05:59    13-Nov-87          Page 65
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0021 *          integer  stype(nwmax)      ! surface type wing wake or tunnel
     *  c                                    >0 NACA 4 digit airfoil
     *  c                                    =0 NACA 64A005
     *  c                                    = -10 tunnel
     *  c
0022 *          integer  nc(nwmax),        ! wing chordwise panels
     *          2        nct(nwmax),       ! total chordwise panels
     *          2        ns(nwmax),        ! spanwise panels
     *          3       !bc(nwmax),        ! boundary condition
     *          4        nsep(nwmax,nsmax)! separation pt
     *  c                separates at nsep panels upstream of trailing edge
0023 *          integer  itws,             ! wake shape iterations
     *          6        itsp,             ! separation pt iterations
     *          7        itto              ! total iterations
     *
     *
0024 *          common /lwingl/ nw, ign, ipn, lal, la2, sw, cwak, cw, x0, z0,
     *          1               alpha, dihed, eh, gbh, reyn, stype,
     *          2               nc, nct, ns, nsep, itws, itsp, itto
     *
0025           include 'lwing2.var'
     *  c       file 'lwing2.var'
     *  c       changed from common block to allow use of virtual memory
     *
0026 *          integer iga,
     *          1       iul,
     *          2       inc,
     *          3       iwng
0027 *          real    la,
     *          1       theta,
     *          2       chi,
     *          3       c,
     *          4       b,
     *          5       xp, yp, zp,
     *          6       xc, yc, zc
     *
0028 *         virtual    iga(pmax), iul(pmax), inc(pmax), iwng(pmax),
     *        1           la(pmax), theta(pmax), chi(pmax), c(pmax), b(pmax),
     *        2           xp(pmax), yp(pmax), zp(pmax),
     *        3           xc(pmax), yc(pmax), zc(pmax)
     *
0029           include 'lhist.com'
     *  c       file [20,64]lhist.com
     *  c       Used to keep information for convergence history of method
     *
     *
0030 *          real    cl(nwmax,itmax) !cl vs iteration history
     *
0031 *          common /lhist/ cl
     *
     *


0032           real      clip(pmax), cdip(pmax)
0033           logical   finish, llblt
0034           integer   i,j
```

```
0035          integer   iunit
0036          character*10      fname
0037          character*8       stime    ! start time
0038          character*8       ctime    ! current time
0039          character*9       cdate    ! current date

0040          integer   ipi
0041          virtual   ipi(nwmax,2,nsmax,ncmax)

0042          real              CLg
0043          virtual           CLg(nwmax,0:nsmax)
0044          real              capgam(nwmax,nsmax)
0045          real              cll(nwmax,nsmax)
0046          real              clfrac(nwmax,nsmax)
0047          real              strip    ! area of a chordwise strip of wing
        c      real             cl(nwmax,itmax)
0048          real              cd(nwmax)
0049          real              cllo(nwmax,nsmax)
0050          real              clfr(nwmax,nsmax)
0051          real              cdlo(nwmax,nsmax)
0052          real              cdfr(nwmax,nsmax)
0053          real              cp

0054          real              la3(nwmax)       ! tan of sweep of 1/4 chord
0055          real              ar(nwmax)        ! AR of wing
0056          real              taper(nwmax)     ! taper of wing

0057          integer           iull,ins,incl

0058          real      u(pmax),
                1       v(pmax),
                2       w(pmax),
                3       ul(pmax),
                4       vl(pmax),
                5       wl(pmax)

0059          real      bpan,a

0060          real      pi
0061          real      gamma(pmax)      ! gamma at node i
0062          real      s(nwmax)         ! area of full wing
0063          integer ip0,
                1       ip,
                2       inw
0064          real      sum
0065          real      dely,
                1       gle,
                2       gte,
                3       chord

0066          pi = acos(-1.0)
0067          call date(cdate)
0068          call time(ctime)
```

```
        c       open output file
        c
0069            if (llblt) then
0070            iunit = 1
0071            close(iunit)
0072            fname(1:7) = 'lfly3f.'
0073            fname(8:10) = run(1:3)
0074            open (unit=iunit,name=fname,status='new',dispose='keep')


        c       output header to file
        c
0075            write(iunit,*) '    '
0076            write(iunit,501)    'File LFLY3F.', run(1:3),
                  1       cdate, stime, ctime
0077    501     format( 1x,2a,10x,a,8x,a,8x,a)

0078            write(iunit,'(1x,2a)')    'Title             ', title(1:40)
0079            write(iunit,'(1x,a,i2)') 'Number of wings  ', nw
0080            if (.not.finish) then
0081              write(iunit,'(1x,a,i2)') 'Iteration        ', itto
0082            else
0083              write(iunit,'(1x,a,i2,a)') 'Iteration        ', itto ,
               1                                '  solution converged '
0084            endif

0085            write(iunit,'(1x,a,f6.2)') 'Reynolds # (M)    ', reyn*1.e-6
0086            write(iunit,*) '    '

0087            end if

        c       find the lift CLgamma of each wing
        c       find the area, AR, sweep of 1/4 chord and
        c       taper of each wing


0088            do 102 inw = 1,nw
0089              cl(inw,itto) = 0
0090              cd(inw) = 0
0091              clg(inw,0) = 0

0092              a = la1(inw) - la2(inw)
0093              la3(inw)  =  la1(inw) - 1./4.*a ! tan of sweep of 1/4 chord line
        c       s = total area of wing
        c       projected area = s*cos(dihed(inw))
0094              s(inw) = 2 * sw(inw) * (cw(inw)-0.5*sw(inw)*a)
0095              ar(inw) = (2*sw(inw))**2/s(inw)
0096              taper(inw) = ( cw(inw) - a*sw(inw) )/ cw(inw)


0097              do 112 ins = 1,ns(inw)
0098                capgam(inw,ins) = 0
0099                cllo(inw,ins) = 0
0100                clfr(inw,ins) = 0
0101                cdfr(inw,ins) = 0
```

```
0102              do 122 iul1 = 1,2
0103                sum = 0
0104                do 132 incl = 1,nc(inw)
                c-----------------------------------------
0105            ip    = ipi(inw,iul1,ins,incl)
0106            i     = iga(ip)
0107            gle   = gamma(i)
0108            gte   = gamma(i+1)
0109            chord = c       (ip)
0110            dely  = b       (ip)*cos(chi(ip))

0111            cp    = 1. - ul(ip)**2 - vl(ip)**2

0112            cl(inw,itto) = cl(inw,itto) - 2./s(inw)*cp*clip(ip)
0113            cd(inw)      = cd(inw)      + 2./s(inw)*cp*cdip(ip)

0114            cllo(inw,ins) =  cllo(inw,ins) -           cp*clip(ip)
0115            clfr(inw,ins) =  clfr(inw,ins) - 2./s(inw)*cp*clip(ip)

0116            cdlo(inw,ins) =  cdlo(inw,ins) +           cp*cdip(ip)
0117            cdfr(inw,ins) =  cdfr(inw,ins) + 2./s(inw)*cp*cdip(ip)

0118            sum = sum + 2*dely*chord*(gle+gte)/s(inw)
0119            capgam(inw,ins) = capgam(inw,ins) + chord*(gle+gte)/2.0

                c-----------------------------------------
0120        132         continue
0121                    if (iul1.eq.1) clg(inw,ins) = sum
0122                    if (iul1.eq.2) clg(inw,ins) = clg(inw,ins) + sum
0123        122       continue
0124                  clg(inw,0) = clg(inw,0) + clg(inw,ins)

0125                  strip = sw(inw)/ns(inw)
                    1   *  (cw(inw) - a*(ins-0.5)/ns(inw)*sw(inw) )

0126                  cllo(inw,ins) = cllo(inw,ins)/strip
0127                  cdlo(inw,ins) = cdlo(inw,ins)/strip

0128                  cll(inw,ins) = clg(inw,ins)*s(inw)/(2*strip)
0129        112     continue ! ins
0130        102   continue ! inw


                c       out put information for each wing
                c
                c
0131            if (llblt) then

0132            do 103 inw = 1,nw
0133              write(iunit,*)     '    '
0134              write(iunit,'(a,i2,i6)')    ' Wing number, type   ',inw,stype(inw)
0135              write(iunit,*)     '    '

0136              write(iunit,'(2a)')
                  1  '  alpha  ns   nc ncwake   AR    taper  sweep(1/4c)',
```

```
                2  ' wake length '

0137            write(iunit,'(f7.2,i4,i5,i6,f8.2,f8.2,f9.2,f10.2)')
                1  alpha(inw)*180/pi, ns(inw), nc(inw), nct(inw)-nc(inw),
                1  ar(inw),
                2  taper(inw), atan(la3(inw))*180/pi, cwak(inw)

0138            write(iunit,*)    '   '

0139            write(iunit,'(2a)')
                1  '  x0      z0     dehedral omega(le) omega(te)   c ',
                2  '   b/2      area  '

0140            write(iunit,'(2f8.4,f7.2,f9.2,f10.2,f8.2,2f8.2)')
                1  x0(inw), z0(inw), dihed(inw)*180/pi,
                2  atan(la1(inw))*180/pi, atan(la2(inw))*180/pi, cw(inw),
                3  sw(inw), s(inw)

0141            write(iunit,*) '  '

0142            write(iunit,'(2(a,f8.4))') ' CL =  ', cl(inw,itto),
              1                            '     CD =  ', cd(inw)



0143            write(iunit,'(a,f8.4)') ' CLgamma   ',clg(inw,0)
0144            write(iunit,*) '   '

0145            write(iunit,'(2a)') '   ins   Capgam    Clg(local)  CLgfrac ',
              1                     '   cl      cd      clfrac      cdfrac  '

0146            do 113 ins = 1,ns(inw)
0147              write(iunit,502) ' ',
              1   ins, capgam(inw,ins), cll(inw,ins), clg(inw,ins),
              1   cllo(inw,ins), cdlo(inw,ins), clfr(inw,ins), cdfr(inw,ins)

0148    502       format (a,i4,7f10.4)

0149    113     continue
0150            write(iunit,*) '   '
0151            write(iunit,*) '   '
0152            write(iunit,'(3a)') '  iul ins inc  ip   gle     gte     ul',
                1  '      vl     wl       cp     lamda    theta    chi     c ',
                2  '      b '

0153            write(iunit,*) '   '

        c       output the panel values
        c
        c
        c
0154            do 114 ins = 1,ns(inw)
0155              do 123 iull = 1,2
0156                do 133 incl = 1,nct(inw)
0157                  ip = ipi(inw,iull,ins,incl)
```

```
PDP-11 FORTRAN-77 V5.0-0        15:05:59    13-Nov-87          Page 70
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0158                  write(iunit,5011)
                1         iul1, ins, incl, ip, gamma(iga(ip)), gamma(iga(ip)+1),
                2         ul(ip), vl(ip), wl(ip),
                3         (1.0 - ul(ip)**2 - vl(ip)**2 ),
                4         la(ip), theta(ip), chi(ip), c(ip), b(ip)
0159    504     format(3i4,4x,f8.4)
0160    5011    format(4i4,20f8.4)

0161    133         continue
0162    123       continue
0163    114     continue
0164    103   continue
0165          close(1)

0166          end if

0167          call lclout
0168          call lcpout (ipi,ul,vl)
0169          call lshout (ipi,
                1            iga, iul, inc, iwng,
                2            la, theta, chi, c, b,
                3            xp, yp, zp, xc, yc, zc )

0170          end


PROGRAM SECTIONS

 Name       Size                  Attributes

$CODE1  006464  1690            RW,I,CON,LCL
$PDATA  001254   342            RW,D,CON,LCL
$IDATA  000454   150            RW,D,CON,LCL
$VARS   000356   119            RW,D,CON,LCL
$TEMPS  000034    14            RW,D,CON,LCL
LCHAR   000054    22            RW,D,OVR,GBL
LWING1  000170    60            RW,D,OVR,GBL
LHIST   000050    20            RW,D,OVR,GBL


Total Space Allocated = 011342  2417

Total Virtual Array Storage = 1
```

```
PDP-11 FORTRAN-77 V5.0-0        15:06:28    13-Nov-87          Page 71
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0001          subroutine lbc(  qlw, qu, qw, gamma,
                1               iga, iul, inc, iwng,
                2               la, theta, chi, c, b,
                3               xp, yp, zp, xc, yc, zc    )


        c       input: qlw is set for 0 normal velocity for rows
        c              1 to ipn, gamma set to rhs
        c       input: qlw is set to 0 for rows below ipn
        c              gamma == 0 for rows below ipn
        c
        c       output: the boundary conditions have been set
        c               and the matrix can be solved to find the
        c               velocities
        c               qlw: coeficient  matrix
        c               gamma: right hand side of matrix equation
        c

0002          implicit character*200 (a-z)

0003          include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0004 *          integer    pmax
0005 *          parameter (pmax = 90) ! max number of panels
0006 *          integer    nwmax
0007 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0008 *          integer    nsmax
0009 *          parameter (nsmax=2)  ! max number of spanwise stations
0010 *          integer    ncmax
0011 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0012 *          integer    itmax
0013 *          parameter (itmax=5) ! max number of total iterations
0014          include 'lwing1.com'
     *  c       file [20,64]lwing1.com
     *
0015 *          integer  nw,                ! number of surfaces
     *          1                           ! wing + wake(s), & tunnel
     *          2        ign,               ! number of gammas
     *          3        ipn                ! number of panels
0016 *          real     la1(nwmax),
     *          1        la2(nwmax),
     *          2        sw(nwmax),         ! half span = b/2
     *          2        cwak(nwmax),       ! length of wake
     *          3        cw(nwmax),         ! chord of wing
     *          4        x0(nwmax),
     *          5        z0(nwmax),
     *          6        alpha(nwmax),
     *          7        dihed(nwmax),
     *          8        eh,                ! tunnel ellipse height
     *          9        gbh,               ! tunnel ground board height
     *          1        reyn               ! reynolds number e-6
0017 *          integer  stype(nwmax)       ! surface type wing wake or tunnel
     *  c                                     >0 NACA 4 digit airfoil
     *  c                                     =0 NACA 64A005
```

```
PDP-11 FORTRAN-77 V5.0-0        15:06:28    13-Nov-87          Page 72
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

     *  c                                      = -10 tunnel
     *  c
0018 *          integer  nc(nwmax),        ! wing chordwise panels
     *          2        nct(nwmax),       ! total chordwise panels
     *          2        ns(nwmax),        ! spanwise panels
     *          3       !bc(nwmax),        ! boundary condition
     *          4        nsep(nwmax,nsmax)! separation pt
     *  c                separates at nsep panels upstream of trailing edge
0019 *          integer  itws,             ! wake shape iterations
     *          6        itsp,             ! separation pt iterations
     *          7        itto              ! total iterations
     *
     *
0020 *          common /lwing1/ nw, ign, ipn, la1, la2, sw, cwak, cw, x0, z0,
     *          1               alpha, dihed, eh, gbh, reyn, stype,
     *          2               nc, nct, ns, nsep, itws, itsp, itto
     *
0021           include 'lwing2.var'
     *  c       file 'lwing2.var'
     *  c       changed from common block to allow use of virtual memory
     *
0022 *          integer iga,
     *          1       iul,
     *          2       inc,
     *          3       iwng
0023 *          real    la,
     *          1       theta,
     *          2       chi,
     *          3       c,
     *          4       b,
     *          5       xp, yp, zp,
     *          6       xc, yc, zc
     *
0024 *         virtual    iga(pmax), iul(pmax), inc(pmax), iwng(pmax),
     *        1           la(pmax), theta(pmax), chi(pmax), c(pmax), b(pmax),
     *        2           xp(pmax), yp(pmax), zp(pmax),
     *        3           xc(pmax), yc(pmax), zc(pmax)
     *

0025           real      qlw      ! matrix to be solved
0026           real      qu,      ! u global coeffienct matrix
                 1       qw       ! w global coeffienct matrix

0027           virtual qlw(pmax,pmax),
                 1       qu(pmax,pmax),
                 2       qw(pmax,pmax)

0028           real      qlu              ! u local coeffient
0029           real      gamma(pmax)      ! right hand side of matrix


0030           integer  neql, ig, ip, ic, ipw, ipwl
0031           integer  ip0,ip1,ip2,ip3,ip4,ip5,ip6,ip7,ip8
0032           integer inw,ins
```

```
PDP-11 FORTRAN-77 V5.0-0        15:06:28    13-Nov-87          Page 73
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0033          real sth,cth,sch,cch
0034          real pi

0035          pi = acos(-1.0)

        c       first unused equation, 2 needed for each spanwise strip

0036          neql = ipn + 1
0037          ip0 = 0
0038          do 101 inw = 1,nw  ! for each wing
        c         ip0 = the last panel before this wing
0039            if (inw.ne.1) ip0 = ip0 + 2*nct(inw-1)*ns(inw-1)
0040            do 111 ins = 1,ns(inw) ! for each spanwise strip

        c           panel index numbers for this spanwise station
        c           array iga is used to get gamma numbers from panel numbers

0041              ipl = ip0 + 1 + nct(inw)*(ins-1) ! top 1st panel of wing
0042              ip2 = ipl + nct(inw)*ns(inw)     ! bottem 1st panel of wing

0043              ip3 = ip0 + nct(inw)*(ins-1) + nc(inw) ! top last panel of wing
0044              ip4 = ip3 + nct(inw)*ns(inw)     ! bottem last panel of wing

0045              ip5 = ip3 + 1 - nsep(inw,ins) ! top 1st panel of wake
0046              ip6 = ip4 + 1           ! bottem 1st panel of wake
0047              ip7 = ip0 + nct(inw)*ins        ! top last panel of wake
0048              ip8 = ip7 + nct(inw)*ns(inw)  ! bottem last panel of wake


0049              if (stype(inw).eq.-10) then  ! tunnel
        c             boundary conditions for tunnel
        c             gamma trailing edge = 0.0
0050                qlw(neql,iga(ip3)+1) = 1.0     ! gamma(upper te) = 0
0051                neql = neql + 1
0052                qlw(neql,iga(ip4)+1) = 1.0     ! gamma(lower te) = 0
0053                neql = neql + 1

0054              else


0055                qlw(neql,iga(ipl)) =  1.0     ! leading edge condition
0056                qlw(neql,iga(ip2)) = -1.0     ! equal
0057                neql = neql + 1


0058                if (nsep(inw,ins).eq.0) then ! separation test------------------

        c               attached flow at this strip
        c               kutta condition
        c               trailing edge velocity extrapolated from last two panels
        c               Vt(te upper) - Vt(te lower) = 0


        c               last top panel
0059                  ic = ip3
```

```
0060                  sth = sin(theta(ic))
0061                  cth = cos(theta(ic))
0062                  sch = sin(chi(ic))
0063                  cch = cos(chi(ic))

0064                  do 122 ig = 1,ign
0065                    qlu =  qu(ic,ig)*cth   + qw(ic,ig)*sth
0066                    qlw(neql,ig) = qlu
0067      122         continue
0068                  gamma(neql) = (-4*pi*cth)

          c             trailing edge bottem panel
0069                  ic = ip4
0070                  sth = sin(theta(ic))
0071                  cth = cos(theta(ic))
0072                  sch = sin(chi(ic))
0073                  cch = cos(chi(ic))

0074                  do 124  ig = 1,ign
0075                    qlu =  qu(ic,ig)*cth   + qw(ic,ig)*sth
0076                    qlw(neql,ig) = qlw(neql,ig) - qlu

          c               zero the equations set for Vn upper and lower
          c               trailing edge = 0
          c               then set up gamma trailing edge = 0 instead
0077                    qlw(ip3,ig) = 0
0078                    qlw(ip4,ig) = 0
0079      124         continue

0080                  gamma(neql) = gamma(neql) -  (-4*pi*cth)

          c           gamma trailing edge = 0
0081                  gamma(ip3) = 0
0082                  gamma(ip4) = 0

          c           set the trailing edge vorticity = 0
0083                  qlw(ip3,iga(ip3)+1) = 1.0 ! upper
0084                  qlw(ip4,iga(ip4)+1) = 1.0 ! lower

0085                  neql = neql + 1


          c       set all of the wake values of gamma to zero
          c       write over Vn=0 for wake

          c       upper surface
0086                  do 125 ipw = ip5, ip7 ! upper wake
0087                    do 131 ig = 1, ign
0088                      qlw(ipw,ig) = 0.0
0089      131             continue ! end do
0090                    gamma(ipw) = 0.0
0091                    qlw(ipw,iga(ipw)+1) = 1.0
0092      125           continue ! end do
```

```
PDP-11 FORTRAN-77 V5.0-0        15:06:28    13-Nov-87           Page 75
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

        c       lower surface
0093                  do 126 ipw = ip6,ip8
0094                    do 132 ig = 1, ign
0095                      qlw(ipw,ig) = 0.0
0096    132             continue ! end do
0097                    gamma(ipw) = 0.0
0098                    qlw(ipw,iga(ipw)+1) = 1.0
0099    126           continue ! end do


0100                else ! separated flow ------------------------------------


        c       constant strength trailing vortex sheets
        c       first set sheet strength = and opposite

0101                  qlw(neql,iga(ip5)) = 1.0
0102                  qlw(neql,iga(ip6)) = 1.0
0103                  neql = neql + 1

        c       use the equations for the control points to set the rest of the
        c       gammas

0104                  do 701 ic = ip5, ip7
0105                    do 702 ig = 1,ign
0106                      qlw(ic,ig) = 0
0107    702             continue
0108                    gamma(ic) = 0
0109                    qlw(ic, iga(ic )+1) =  1.0
0110                    qlw(ic, iga(ip5)  ) = -0.5
0111                    qlw(ic, iga(ip6)  ) =  0.5
0112    701           continue

0113                  do 703 ic = ip6, ip8
0114                    do 704 ig = 1,ign
0115                      qlw(ic,ig) = 0
0116    704             continue
0117                    gamma(ic) = 0

0118                    qlw(ic, iga(ic )+1) =  1.0
0119                    qlw(ic, iga(ip5)  ) =  0.5
0120                    qlw(ic, iga(ip6)  ) = -0.5

0121    703           continue

0122                end if  ! separation ? -----------------------------------
0123              end if

0124    111     continue
0125    101   continue

        d       if ( (neql-1).ne.ign )  then
        d         type *, ' neql = ',neql, 'non square system '
        d        call exit
        d       end if
```

```
0126          return
0127          end
```

PROGRAM SECTIONS

Name	Size		Attributes
$CODE1	004004	1026	RW,I,CON,LCL
$PDATA	000010	4	RW,D,CON,LCL
$IDATA	000276	95	RW,D,CON,LCL
$VARS	000072	29	RW,D,CON,LCL
$TEMPS	000020	8	RW,D,CON,LCL
LWING1	000170	60	RW,D,OVR,GBL

Total Space Allocated = 004614 1222

```
PDP-11 FORTRAN-77 V5.0-0        15:06:46    13-Nov-87          Page 77
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0001          subroutine lngeom ( u, v, w, ul, vl, wl,
                2                   iga, iul, inc, iwng,
                3                   la, theta, chi, c, b,
                4                   xp, yp, zp, xc, yc, zc, nsepol  )

        c
        c       need to keep track of last panel inorder to get a translation
        c       for the next panel to make the wake continues
        c
        c
        c

        c       subroutine L_new_geometry

        c       input
        c       the velocities have been computed
        c

        c       output
        c       does all wings
        c       a new geometry file has been written to lfly2u.ftn
        c       and to lfly2f.ftn
        c       or
        c       if the solution is finished the flag finish is set to true
        c

0002          implicit character*200 (a-z)

0003          include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0004 *          integer    pmax
0005 *          parameter (pmax = 90) ! max number of panels
0006 *          integer    nwmax
0007 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0008 *          integer    nsmax
0009 *          parameter (nsmax=2)  ! max number of spanwise stations
0010 *          integer    ncmax
0011 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0012 *          integer    itmax
0013 *          parameter (itmax=5) ! max number of total iterations
0014          include 'lchar.com'
     *  c       file [20,64]lchar.com
     *
0015 *          character*40  title
0016 *          character*4  run
     *
0017 *          common /lchar/  title,run
     *
0018          include 'lwingl.com'
     *  c       file [20,64]lwingl.com
     *
0019 *          integer  nw,              ! number of surfaces
     *          1                         ! wing + wake(s), & tunnel
     *          2        ign,             ! number of gammas
```

```
     *          3         ipn                ! number of panels
0020 *          real      la1(nwmax),
     *          1         la2(nwmax),
     *          2         sw(nwmax),         ! half span = b/2
     *          2         cwak(nwmax),       ! length of wake
     *          3         cw(nwmax),         ! chord of wing
     *          4         x0(nwmax),
     *          5         z0(nwmax),
     *          6         alpha(nwmax),
     *          7         dihed(nwmax),
     *          8         eh,                ! tunnel ellipse height
     *          9         gbh,               ! tunnel ground board height
     *          1         reyn               ! reynolds number e-6
0021 *          integer   stype(nwmax)       ! surface type wing wake or tunnel
     *  c                                      >0 NACA 4 digit airfoil
     *  c                                      =0 NACA 64A005
     *  c                                      = -10 tunnel
     *  c
0022 *          integer   nc(nwmax),         ! wing chordwise panels
     *          2         nct(nwmax),        ! total chordwise panels
     *          2         ns(nwmax),         ! spanwise panels
     *          3        !bc(nwmax),         ! boundary condition
     *          4         nsep(nwmax,nsmax)! separation pt
     *  c                  separates at nsep panels upstream of trailing edge
0023 *          integer   itws,              ! wake shape iterations
     *          6         itsp,              ! separation pt iterations
     *          7         itto               ! total iterations
     *
     *
0024 *          common /lwing1/ nw, ign, ipn, la1, la2, sw, cwak, cw, x0, z0,
     *          1                alpha, dihed, eh, gbh, reyn, stype,
     *          2                nc, nct, ns, nsep, itws, itsp, itto
     *
0025          include 'lwing2.var'
     *  c       file 'lwing2.var'
     *  c       changed from common block to allow use of virtual memory
     *
0026 *          integer iga,
     *          1       iul,
     *          2       inc,
     *          3       iwng
0027 *          real    la,
     *          1       theta,
     *          2       chi,
     *          3       c,
     *          4       b,
     *          5       xp, yp, zp,
     *          6       xc, yc, zc
     *
0028 *        virtual    iga(pmax), iul(pmax), inc(pmax), iwng(pmax),
     *       1           la(pmax), theta(pmax), chi(pmax), c(pmax), b(pmax),
     *       2           xp(pmax), yp(pmax), zp(pmax),
     *       3           xc(pmax), yc(pmax), zc(pmax)
     *
```

```
PDP-11 FORTRAN-77 V5.0-0        15:06:46    13-Nov-87          Page 79
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0029          character*10      fname   ! file name lfly2(u or f).xxx

0030          integer   nsepol(nwmax,nsmax)

0031          real      u(pmax),   v(pmax),   w(pmax),
                1       ul(pmax),  vl(pmax),  wl(pmax)

0032          integer ip0,ip1,ip2,ip3,ip4,ip5,ip6,ip7,ip8

0033          real      pi

0034          real      x1(4),y1(4),z1(4)  ! wing ref frame
0035          real      x2(4),y2(4),z2(4)  ! global ref frame
0036          real      x3(4),y3(4),z3(4)  ! panel ref frame

0037          integer incl,       ! chordwise paneling counter, wing
                1       inc2,       ! chordwise paneling counter, wake
                1       ins,        ! spanwise paneling counter
                2       inw,        ! wing counter
                3       iunit,
                4       iull,       ! surface counter
                5       ipt,ip

0038          real      sth,cth,cch,sch
0039          real      xt,yt,zt
0040          real      xp3
0041          real      xoff    ! off set of upstream panel
0042          real      zoff    ! used to make a continious wake

0043          real      arot    ! angle of rotation of wake panel

0044          real      rotmax  ! maximum angle of rotation

0045          integer ic,
                1       ig

0046          integer   i,j,k

0047          real      eps     ! maximum velocity through wake

      c-------------------------------------------------------
      c-------------------------------------------------------
      c       end declairation and start code
      c-------------------------------------------------------
      c-------------------------------------------------------


      c
      c       to local from global
      c       xl =   x*cth                    + z*sth
      c       yl = - x*sth*sch + y*cch        + z*cth*sch
      c       zl = - x*sth*cch - y*sch        + z*cth*cch
      c
      c       to global from local
      c       x =   xl*cth     - yl*sch*sth - zl*cch*sth
```

```
PDP-11 FORTRAN-77 V5.0-0        15:06:46    13-Nov-87          Page 80
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

        c       y =                 yl*cch     - zl*sch
        c       z =   xl*sth     + yl*sch*cth + zl*cch*cth

0048            write(3,'(a,i4)')  ' lngeom        itto = ', itto
0049            pi = acos(-1.0)

0050            eps = 0.005

        c         rotation maximun allowed
        c         f(alpha,ip)

0051            if (itto.eq.1) then
0052              rotmax = 20.0
0053            else
0054              rotmax = 10.0
0055            end if
0056            iunit = 1

        c           open file 'lfly2u.xxx' for new panel geometry

        d           close(1)
        c           close(2)

        d           fname(1:7) = 'lflylf.'  ! formatted - can be typed out

0057            fname(8:10) = run(1:3)
        d           open(unit=1,name=fname,status='new',dispose='keep')

0058            fname(1:7) = 'lflylu.'  ! unformatted - input for lfly2

0059            open(unit=2,name=fname,form='unformatted',
                  1     status='new',dispose='keep')

        d           write(3,*) ' opening files ',fname

        d           write(1,'(3a)')
        d         1  '  ip  iga iul inc iwng      la    theta      chi        c  ',
        d         2  '         b        xp       yp       zp        xc       yc      zc

        c           initialize panel counters
0060            ip0 = 0
0061            ip  = 0

0062            do 102 inw = 1,nw  ! for each wing

        c             ip0 = the last panel before this wing
0063              if (inw.ne.1) ip0 = ip0 + 2*nct(inw-1)*ns(inw-1)

0064              do 111 iul1 = 1,2                       ! upper/lower surface
0065                do 121 ins = 1,ns(inw)                ! span divisions
0066                  do 131 incl = 1,nct(inw)       ! chord divisions

        c         change indentation to keep from running off of the page
        c<<<<<<<<<<<<<<<<< ////////////////////////////////////////////
```

```
0067          ip = ip + 1

        c       define ip1,...ip8
        c       panel index numbers for this spanwise station

0068          ip1 = ip0 + 1 + nct(inw)*(ins-1) ! top 1st panel of wing
0069          ip2 = ip1 + nct(inw)*ns(inw)     ! bottem 1st panel of wing
0070          ip3 = ip0 + nct(inw)*(ins-1) + nc(inw) ! top last panel of wing
0071          ip4 = ip3 + nct(inw)*ns(inw)     ! bottem last panel of wing

0072          ip5 = ip3 + 1 - nsep(inw,ins) ! top 1st panel of wake
0073          ip6 = ip4 + 1                 ! bottem 1st panel of wake
0074          ip7 = ip0 + nct(inw)*ins      ! top last panel of wake
0075          ip8 = ip7 + nct(inw)*ns(inw)  ! bottem last panel of wake


        c       test for panels that need to be modified
        c       wing panels and tunnel panels are left alone
        c       then change those panels
        c       output other panels without change
        c

0076          if (     (    (ip6.le.ip.and.ip.le.ip8)
                1          .or.(ip5.le.ip.and.ip.le.ip7))
                2    .and.(stype(inw).ne.-10)             )  then


        c         1) set up panel coordinates
        c         2) rotate and translate coordinate system into global coordinates
        c         3) rotate panel to new wake shape
        c         4) find local panel parameters


        c         define the panel in local coordinates x3,y3,z3 (4 corners)
        c
0077            x3(1) = 0
0078            y3(1) = 0
0079            z3(1) = 0

0080            x3(2) = c(ip)
0081            y3(2) = 0
0082            z3(2) = 0

0083            x3(3) = la(ip)*b(ip)
0084            y3(3) = b(ip)
0085            z3(3) = 0

0086            x3(4) = x3(3) + c(ip)
0087            y3(4) = b(ip)
0088            z3(4) = 0


        c         rotate and translate into global frame
```

```
0089            sth = sin(theta(ip))
0090            cth = cos(theta(ip))
0091            sch = sin(chi(ip))
0092            cch = cos(chi(ip))

        c         to global from local
        c         x =   xl*cth     - yl*sch*sth - zl*cch*sth
        c         y =                yl*cch     - zl*sch
        c         z =   xl*sth     + yl*sch*cth + zl*cch*cth

0093            if (ip.eq.ip5.or.ip.eq.ip6) then ! start of wake sheet
0094              xoff = xp(ip)
0095              zoff = zp(ip)
0096            end if


0097            do 141 i = 1,4
0098              x2(i) =   x3(i) *cth     - y3(i) *sch*sth - z3(i) *cch*sth
                1       +   xoff

0099              y2(i) =                    y3(i) *cch     - z3(i) *sch
                1       +   yp(ip)

0100              z2(i) =   x3(i) *sth     + y3(i) *sch*cth + z3(i) *cch*cth
                1       +   zoff
0101    141     continue

        c         define the new coordinates in this frame for the rotated wake
        c         change to trailing edge position of this panel



0102            arot = atan( wl(ip)/ul(ip) )

0103            write(3,'(a,i6,2f8.2)')
              1  ' ip arot  = ', ip, arot*180./pi, theta(ip)*180./pi


        c         0.95 work well for the unseparated case

        c         typical rotation correction
        c         arot = arot * 0.95


0104            if (nsep(inw,ins).eq.0) then
0105              arot = arot * 0.950
0106            else
0107              arot = arot * 1.5
0108            end if

0109            if (abs(arot).gt.rotmax*pi/180.) then
0110              arot = sign(rotmax*pi/180.0, arot)
0111              type *,' max angle used = ', arot*180./pi
0112              write(3,*) ' max angle used = ', arot*180./pi
0113            end if
```

```
0114            theta(ip) = theta(ip) + arot

0115            if (itws.ne.1) then
0116              if (ip.lt.ip5+nsep(inw,ins)-nsepol(inw,ins) ) then
0117                theta(ip) = -alpha(inw)*0.5
0118              end if
0119            end if


0120            x2(2)   = x2(1) + c(ip)*cos(theta(ip))
0121            z2(2)   = z2(1) + c(ip)*sin(theta(ip))

        c         store position of trailing edge for the next panel
        c         to match this one

0122            xoff = x2(2)
0123            zoff = z2(2)

0124            x2(4)   = x2(3) + c(ip)*cos(theta(ip))
0125            z2(4) = z2(3) + c(ip)*sin(theta(ip))

        c         find theta, and chi such that z1 = zero for all points
        c         to local from global
        c         x1 =   x*cth                    + z*sth
        c         y1 = - x*sth*sch + y*cch        + z*cth*sch
        c         z1 = - x*sth*cch - y*sch        + z*cth*cch

0126            theta(ip) = atan2( (z2(2)-z2(1)), (x2(2)-x2(1)) )

0127            chi(ip) = atan2( (z2(3)-z2(1))*cos(theta(ip))-
                1                  (x2(3)-x2(1))*sin(theta(ip)) ,
                1                 y2(3)-y2(1)  )

0128            sth = sin(theta(ip))
0129            cth = cos(theta(ip))
0130            sch = sin(chi(ip))
0131            cch = cos(chi(ip))

        c         change to local coordinates to find the panel properties
        c         find corner points in panel system (x3,y3,z3)  (z3 == 0)
        c         translate axis and rotate into panel system

0132            do 142 ipt = 1,4

0133              xt = x2(ipt) - x2(1)
0134              yt = y2(ipt) - y2(1)
0135              zt = z2(ipt) - z2(1)

        c         new local panel system  z3==0 by def.
        c
0136              x3(ipt) =   xt*cth                + zt*sth
0137              y3(ipt) = - xt*sth*sch + yt*cch   + zt*cth*sch
0138              z3(ipt) = - xt*sth*cch - yt*sch   + zt*cth*cch
```

```
0139    142     continue

0140            b(ip) = y3(3)

        c         approximate the swept tapered panel by a swept panel

0141            la(ip) =( x3(3)        /b(ip) +
                1         (x3(4)-x3(2))/b(ip)   )/2.

0142            c(ip) = (x3(2) + (x3(4)-x3(3)) )/2.

        c       write(3,*) ' new b,la,c  ', b(ip),la(ip),c(ip)


        c         inboard leading edge point of approximate panel in
        c         tapered panel reference frame(3)
        c         yp3,zp3 = 0

0143            xp3   = x3(3)/2. - la(ip)*b(ip)/2.

        c         determine the new pt 1 of the panel in global coordinates
        c         rotate, translate to global coordinates

0144            xp(ip) =  x2(1) + xp3*cth     !- yp3*sch*sth - zp3*cch*sth
0145            yp(ip) =  y2(1)               !+ yp3*cch     - zp3*sch
0146            zp(ip) =  z2(1) + xp3*sth     !+ yp3*sch*cth + zp3*cch*cth

        c         find control point of panel
0147            xc(ip) = (x2(1) + x2(2) + x2(3) + x2(4) )/4.
0148            yc(ip) = (y2(1) + y2(2) + y2(3) + y2(4) )/4.
0149            zc(ip) = (z2(1) + z2(2) + z2(3) + z2(4) )/4.

0150          end if ! (wake)

        c       panel configuration is written out at this point
        c       panels on wing and tunnel panels are left unchanged

        d       write(iunit,502)
        d       1
        d       2  ip, iga(ip), iul(ip), inc(ip), iwng(ip),
        d       3
        d       4  la(ip), theta(ip), chi(ip), c(ip), b(ip),
        d       5
        d       6  xp(ip), yp(ip), zp(ip), xc(ip),yc(ip),zc(ip)
0151    502     format(5i4,5f10.4,x,2(3f9.4,x) )

0152          write(2)
                1
                2  ip, iga(ip), iul(ip), inc(ip), iwng(ip),
                3
                4  la(ip), theta(ip), chi(ip), c(ip), b(ip),
                5
                6  xp(ip), yp(ip), zp(ip), xc(ip),yc(ip),zc(ip)
```

```
PDP-11 FORTRAN-77 V5.0-0        15:06:46    13-Nov-87          Page 85
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

        c       change indentation to original
        c>>>>>>>>>>>>>>> ////////////////////////////////////////
0153    131           continue  ! incl
0154    121         continue    ! ins
0155    111       continue      ! iull
0156    102     continue        ! inw
        d       close(1)
0157            close(2)
0158            end


PROGRAM SECTIONS

 Name        Size                 Attributes

$CODE1   005202  1345            RW,I,CON,LCL
$PDATA   000174    62            RW,D,CON,LCL
$IDATA   000350   116            RW,D,CON,LCL
$VARS    000376   127            RW,D,CON,LCL
$TEMPS   000032    13            RW,D,CON,LCL
LCHAR    000054    22            RW,D,OVR,GBL
LWING1   000170    60            RW,D,OVR,GBL


Total Space Allocated = 006642  1745
```

```
PDP-11 FORTRAN-77 V5.0-0        15:07:14    13-Nov-87          Page 86
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0001          subroutine lblt   ( ul, vl, wl, ipi,
             2                  iga, iul, inc, iwng,
             2                  la, theta, chi, c, b,
             2                  xp, yp, zp, xc, yc, zc,
             2                  finish                  )

0002          implicit character*200 (a-z)

0003          include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0004 *          integer    pmax
0005 *          parameter (pmax = 90) ! max number of panels
0006 *          integer    nwmax
0007 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0008 *          integer    nsmax
0009 *          parameter (nsmax=2)  ! max number of spanwise stations
0010 *          integer    ncmax
0011 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0012 *          integer    itmax
0013 *          parameter (itmax=5) ! max number of total iterations
0014          include 'lchar.com'
     *  c       file [20,64]lchar.com
     *
0015 *          character*40  title
0016 *          character*4  run
     *
0017 *          common /lchar/  title,run
     *
0018          include 'lwingl.com'
     *  c       file [20,64]lwingl.com
     *
0019 *          integer  nw,               ! number of surfaces
     *          1                          ! wing + wake(s), & tunnel
     *          2        ign,              ! number of gammas
     *          3        ipn               ! number of panels
0020 *          real     la1(nwmax),
     *          1        la2(nwmax),
     *          2        sw(nwmax),        ! half span = b/2
     *          2        cwak(nwmax),      ! length of wake
     *          3        cw(nwmax),        ! chord of wing
     *          4        x0(nwmax),
     *          5        z0(nwmax),
     *          6        alpha(nwmax),
     *          7        dihed(nwmax),
     *          8        eh,               ! tunnel ellipse height
     *          9        gbh,              ! tunnel ground board height
     *          1        reyn              ! reynolds number e-6
0021 *          integer  stype(nwmax)      ! surface type wing wake or tunnel
     *  c                                    >0 NACA 4 digit airfoil
     *  c                                    =0 NACA 64A005
     *  c                                    = -10 tunnel
     *  c
0022 *          integer  nc(nwmax),        ! wing chordwise panels
     *          2        nct(nwmax),       ! total chordwise panels
```

```
PDP-11 FORTRAN-77 V5.0-0        15:07:14    13-Nov-87          Page 87
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

     *          2        ns(nwmax),       ! spanwise panels
     *          3       !bc(nwmax),       ! boundary condition
     *          4        nsep(nwmax,nsmax)! separation pt
     *  c                separates at nsep panels upstream of trailing edge
0023 *          integer  itws,            ! wake shape iterations
     *          6        itsp,            ! separation pt iterations
     *          7        itto             ! total iterations
     *
     *
0024 *          common /lwingl/ nw, ign, ipn, lal, la2, sw, cwak, cw, x0, z0,
     *          1               alpha, dihed, eh, gbh, reyn, stype,
     *          2               nc, nct, ns, nsep, itws, itsp, itto
     *
0025           include 'lwing2.var'
     *  c       file 'lwing2.var'
     *  c       changed from common block to allow use of virtual memory
     *
0026 *          integer iga,
     *          1       iul,
     *          2       inc,
     *          3       iwng
0027 *          real    la,
     *          1       theta,
     *          2       chi,
     *          3       c,
     *          4       b,
     *          5       xp, yp, zp,
     *          6       xc, yc, zc
     *
0028 *         virtual    iga(pmax), iul(pmax), inc(pmax), iwng(pmax),
     *        1           la(pmax), theta(pmax), chi(pmax), c(pmax), b(pmax),
     *        2           xp(pmax), yp(pmax), zp(pmax),
     *        3           xc(pmax), yc(pmax), zc(pmax)
     *

0029           real      ul(pmax),
              1          vl(pmax),
              2          wl(pmax)

0030           logical   finish
0031           integer   i, inw, ins
0032           integer   n,
              1          ip, ip1, ip2, ip3, ip4

0033           real      re

0034           integer   ipi
0035           virtual   ipi(nwmax,2,nsmax,ncmax)

0036           real      x (2*ncmax),
              1          y (2*ncmax),
              2          cp(2*ncmax),
              3          delta2

0037           integer   newsep,iturb
```

```
0038          do 101 inw = 1,nw
0039            if (stype(inw).gt.-10.and.reyn.gt.0) then ! not tunnel,not invisid

0040              do 111 ins = 1,ns(inw)

0041                n = 2*nc(inw)
0042                re = reyn

0043                ipl = ipi(inw,1,ins,1)
0044                ip2 = ipi(inw,2,ins,1)
0045                ip3 = ipi(inw,1,ins,nc(inw))
0046                ip4 = ipi(inw,2,ins,nc(inw))

0047                do 121 i = 1, n+2
0048                  if (i.le.nc(inw)) then
0049                    ip = ip4 + 1 - i
0050                    x(i) = xp(ip+1)
0051                    y(i) = zp(ip+1)
0052                    cp(i)= 1.0 - ul(ip)**2 - vl(ip)**2
0053                  else if (i.gt.nc(inw)) then
0054                    ip = ipl - 1  + i-nc(inw)
0055                    x(i) = xp(ip)
0056                    y(i) = zp(ip)
0057                    cp(i)= 1.0 - ul(ip)**2 - vl(ip)**2
0058                  end if
0059    121           continue

0060                call lamsep(x,y,cp,n,cw(1),re,nsep(ins,inw),delta2,iturb)
0061                call lturse(x,y,cp,n,cw(1),re,nsep(ins,inw),delta2,iturb,
           1              newsep)

0062                newsep = 2*nc(inw) + 1  - newsep


0063                finish = (finish .and. (nsep(ins,inw).ge.newsep))
0064                nsep(ins,inw) = max( newsep, nsep(ins,inw) )

0065                write (3,*) ' lblt nsep = ', nsep(ins,inw)


0066    111         continue
0067              end if
0068    101     continue

0069          end
```

PROGRAM SECTIONS

Name	Size		Attributes
$CODE1	001476	415	RW,I,CON,LCL
$PDATA	000016	7	RW,D,CON,LCL
$IDATA	000350	116	RW,D,CON,LCL

$VARS	002076	543	RW,D,CON,LCL
$TEMPS	000012	5	RW,D,CON,LCL
LCHAR	000054	22	RW,D,OVR,GBL
LWING1	000170	60	RW,D,OVR,GBL

Total Space Allocated = 004440 1168

```
0001          subroutine lamsep(x,y,cp,n,c,re,nsep,delta2,iturb)

0002          implicit character*250 (a-z)

0003          include '[20,64]lfly.inc'
     *  c       file [20,64]lfly.inc
0004 *          integer    pmax
0005 *          parameter (pmax = 90) ! max number of panels
0006 *          integer    nwmax
0007 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0008 *          integer    nsmax
0009 *          parameter (nsmax=2)  ! max number of spanwise stations
0010 *          integer    ncmax
0011 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0012 *          integer    itmax
0013 *          parameter (itmax=5) ! max number of total iterations

        c       input
0014            real       x(2*ncmax),      ! node locations defined for 1 to n
               1           y(2*ncmax),
               2           cp(2*ncmax),     ! control pt cp
               3           c,               ! reference chord
               4           re               ! reynolds number
0015            integer    n,               ! total number of panels on wing
               1           nsep

        c       output
0016            integer    iturb
0017            real       delta2

        c       local
0018            real       sp,              ! panel length
               3           cpstag,          ! cp max
               4           reth,
               5           rex,
               6           rex1

0019            real       s,
               1           u1,u2,u3,
               2           ds1,ds2,
               3           ri,
               4           du2ds,
               5           rlamda,
               6           rcinv,
               7           rci

0020            integer    istag,
               2           i,
               3           nsurf,
               6           m

        c       statement function
0021            sp(i) = sqrt( (x(i)-x(i+1))**2 + (y(i)-y(i+1))**2 )
```

```
0022          type*, ' n,c,re = ', n, c, re


0023          cpstag    = -10000.0

        c     This section finds the minium Cp and defines
        c     the stagnation point as the body coordinate
        c     in between the two maxium Cp's.  The value of
        c     Cp at this point will be taken as one.
        c

0024          type *, ' lamsep find cpstag '
0025          do 101 i = 3,n-2
0026            if(cp(i).gt.cpstag) then
0027              cpstag  = cp(i)
0028              istag   = i
0029            endif
0030            if(cp(istag+1).gt.cp(istag-1))  istag = istag + 1
0031    101   continue
0032          write(3,*) ' cpstag, istag = ',cpstag,istag

        CC                                                              CC
        c     SUBROUTINE LAMSEP(X,Y,SP)
0033          TYPE *,' >>> THWAITES'
        c
        c       This subroutine calculates the laminar SEPARATION point
        c       on the airfoil by Thwaites Integral Method.
        c
        c       The separation point calculations are done on the upper surface
        c       begining at the stagnation point and
        c       proceeding to the trailing edge. The counter in the inner
        c       do loop refers to the control point number.  All separation
        c       points are taken to be the upstream end point of the panel
        c       where separation has been determined.  (In other words,
        c       the leading edge of the separation panel.)
        c       Therefore:
        c
        c               Xlsep(upper) = X(ILSEP(1))
        c

        c      ILSEP    = N+1
        c      itrans   = n+1

0034          iturb    = nsep
0035          write(3,*) ' nsep ', nsep

        c
        c       First some variables are intialized,
        c
0036          S         = 0.0
0037          RI        = 0.0

0038          write(3,*) ' 400 i=istag,n+1-nsep ', istag, n+1-nsep
```

```
PDP-11 FORTRAN-77 V5.0-0        15:07:25    13-Nov-87          Page 92
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0039          DO 400 I = istag, n+1 - nsep, 1

        c      DO 400 I = istag, n-1 ,1
        c
        c               This section finds the distances between the control
        c               points, where:
        c                       DS1     = distance between the i-1 and the ith
        c                                       control points
        c                       DS2     = distance between the i and the i+1th
        c                                       control points
        c
0040            DS1         = (SP(I-1)+SP(I))/2.
0041            DS2         = (SP(I)+SP(I+1))/2.

0042          type *,' ds1,ds2 ', ds1,ds2
0043            U1          = SQRT(1.-CP(I-1))
0044            U2          = SQRT(1.-CP(I))
0045            U3          = SQRT(1.-CP(I+1))

0046          type*,' u1-3 = ',u1,u2,u3


0047            IF(I.EQ.istag) THEN
0048              DS1 = SP(istag)/2.
0049              U1  = 0.0
0050            ENDIF

0051            S = S + DS1
        c
        c       Now the derivative of the of the velocity ratio
        c       Ui/Uinf wrt distance along the surface, DUDS
        c
0052            DU2DS = -1.*U1*DS2/(DS1*(DS1+DS2))
                1       + U2*(1./DS1-1./DS2)
                1       + U3*DS1/(DS2*(DS1+DS2))

0053          type *, ' du2ds = ', du2ds
        c
        c       First if the point is too close to the stagnation
        c       point an approximation for Thwaites' lamda is calculated
        c       based on the approximate velocity of a cylinder
        c       with radius equal to the radius of curature.
        c       (see my notes p.13)  RCINV refers to 1/(radius
        c       of curvature).
        c
0054            IF(U2.LE.0.000001)THEN
0055              RCI       = RCINV(I,X,Y)
0056              RLAMDA    = (1.-0.5*(C*S*RCI)**2)/6.
0057              RI        = (S**6)*C*RCI
0058              TYPE*,' RADIUS OF CURVATURE CALLED'
0059            ENDIF

0060            IF(U2.EQ.0.0) THEN
0061              RLAMDA    = 1./6.
0062              RI        = 0.0
```

```
0063            ELSE
        c
        c       If the point is not the stagnation point:
        c       Thwaites' Integral of |(Ui/Uinf)**5 ds is calculated
        c       by the Trapizoid Rule.
        c
0064              RI      = RI+(U2**5+U3**5)*DS2/2.
0065              IF(RI.LT.0.0000001) TYPE*,'  RC NEEDED'
        c
        c                 Finally Thwaites' lamda is calculated and the SEPARATION
        c                 criterion of lamda = -0.2 is applied to detect
        c                 SEPARATION point.
        c
0066              RLAMDA  = DU2DS*RI/(U2**6)
0067            ENDIF


0068            DELTA2 = SQRT(abs(-0.09*C/(RE*DU2DS)))

0069            write(3,'(a,i5,3f9.6)') ' i, delta2, delta2^2, rlamda ',
              1 i, delta2,  -0.09*C/(RE*DU2DS),   rlamda

0070            IF (RLAMDA.LE.-0.2) THEN
0071              DO 1000 M=I+1,N-1
0072                IF(CP(I).GE.CP(M)) GOTO 901
0073    1000        CONTINUE
0074              iturb = I
        C         model the increase in boundary layer due to the separation
        c         bubble before becoming turbulent

        c         DELTA2 = DELTA2*(1./0.65)**(7./4.)

0075              goto 450
0076            ENDIF

0077    901     continue

0078            reth = re *(u2/1.0)*(delta2/c)
0079            rex  = re *(u2/1.0)*(s/c)
0080            rexl = 1.174*(1.+22400./rex)*rex**0.46

0081          type *,' reth, rex, rexl = ',reth,rex,rexl
0082          write(3,*) ' reth, rex, rexl = ',reth,rex,rexl

0083            if ( reth-rexl.gt.0 )  then
        c         transition to turbulence
0084              iturb = i
0085              goto 450
0086            end if

0087    400   CONTINUE

0088          iturb = n+1-nsep
0089          write(3,*) ' fully lam  '
0090    450   continue
```

```
0091          write(3,*) ' lamsep iturb = ',iturb

0092     500  RETURN
0093          END
```

PROGRAM SECTIONS

Name	Size		Attributes
$CODE1	003254	854	RW,I,CON,LCL
$PDATA	000420	136	RW,D,CON,LCL
$IDATA	000056	23	RW,D,CON,LCL
$VARS	000100	32	RW,D,CON,LCL
$TEMPS	000006	3	RW,D,CON,LCL

Total Space Allocated = 004060 1048

```
PDP-11 FORTRAN-77 V5.0-0        15:07:38    13-Nov-87          Page 95
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR


      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CC                                                                  CC
      CC                                                                  CC
0001          REAL FUNCTION RCINV(I,X,Y)
      C
      C DATE: MARCH 3, 1986
      C
0002          include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0003 *            integer    pmax
0004 *            parameter (pmax = 90) ! max number of panels
0005 *            integer    nwmax
0006 *            parameter (nwmax=2)  ! max number of surfaces (wings)
0007 *            integer    nsmax
0008 *            parameter (nsmax=2)  ! max number of spanwise stations
0009 *            integer    ncmax
0010 *            parameter (ncmax=44)  ! max number of chordwise stations
     *  c                                 on a wing including wake
0011 *            integer    itmax
0012 *            parameter (itmax=5) ! max number of total iterations
0013          integer   i
0014          real      x(2*ncmax)
0015          real      y(2*ncmax)
      c
      c       This function calculates the radius of curvature of
      c       the airfoil surface at a particular body coordinate.
      c       The formulas for the derivatives can be found in
      c       Ferziger, Numerical Methods for Engineering Applications
      c       pp.52-53.
      c
0016          H1 = X(I)-X(I-1)
0017          H2 = X(I+1)-X(I)
0018          H3 = H1+H2

0019          DYDX      = Y(I-1)*(-1.*H2/(H1*H3)) + Y(I)*(1./H1-1./H2)
             +                    + Y(I+1)*(H1/(H2*H3))

0020          D2YDX2    = Y(I-1)*(2./(H1*H3)) - Y(I)*(2.*(H1*H2))
             +                    + Y(I+1)*(2./(H2*H3))

0021          RCINV     = SQRT(D2YDX2**2)/((1.+DYDX**2)**1.5)

0022          RETURN
0023          END


PROGRAM SECTIONS

 Name        Size                 Attributes

$CODE1   000406   131             RW,I,CON,LCL
$IDATA   000030    12             RW,D,CON,LCL
$VARS    000024    10             RW,D,CON,LCL
$TEMPS   000010     4             RW,D,CON,LCL
```

Total Space Allocated = 000472 157

```
0001          SUBROUTINE lturse(x,y,cp,n,c,re,nsep,delta2,iturb,itsep)

0002          implicit character*120 (a-z)

        C l_turbulent_separation
        C DATE: APRIL 6, 1986
        c update: AUGUST 24, 1987
        C
        C                TURBULENT BOUNDARY LAYER SEPARATION IS PREDICTED
        C       USING STRATFORD'S CRITERION WITH ONLY ONE EMPIRICAL
        C       PARAMETER  ( B.S. STRAFORD-- "SEPARATION OF THE TURBULENT
        C       BOUNDARY LAYER", NATIONAL GAS TURBINE ESTABLISHMENT,
        C       FARNBOROUGH, ENGLAND JULY 1958) IN THE JOURNAL OF FLUID
        C       MECHANICS VOLUME 5 1959.
        C

0003          include 'lfly.inc'
     *  c        file [20,64]lfly.inc
0004 *           integer    pmax
0005 *           parameter (pmax = 90) ! max number of panels
0006 *           integer    nwmax
0007 *           parameter (nwmax=2)  ! max number of surfaces (wings)
0008 *           integer    nsmax
0009 *           parameter (nsmax=2)  ! max number of spanwise stations
0010 *           integer    ncmax
0011 *           parameter (ncmax=44)  ! max number of chordwise stations
     *  c                                 on a wing including wake
0012 *           integer    itmax
0013 *           parameter (itmax=5) ! max number of total iterations

        c      input
0014            real       x(2*ncmax), y(2*ncmax),
               1           cp(2*ncmax),
               2           c,
               3           re,
               4           delta2

0015            integer    n,
               1           iturb,
               5           nsep

        c      output
0016            integer    itsep


        c      local
0017            integer    i0,
               3           ireat,
               4           istart,
               5           l,i

0018            real       cp0,cp01,cp02,cp03
0019            real       d2cp
```

```
PDP-11 FORTRAN-77 V5.0-0        15:07:43    13-Nov-87          Page 98
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0020          real      sp,
             1          c4,
             3          s0,
             4          s,
             5          cpmin,
             6          fnc,fnc1

0021          real      ds1,ds2,
             1          dcp0ds,
             2          r,
             3          b,
             4          rrhs,
             5          rlhs,
             6          sign

      c      statement function
0022          sp(i) = sqrt( (x(i)-x(i+1))**2 + (y(i)-y(i+1))**2 )


0023          TYPE *, ' >>> TURBSEP'
      c
      c         First find the maxium velocity (minium pressure) within
      c         the turbulent boundary layer.
      c


0024          itsep = n+1 - nsep                ! set itsep to trailing edge


0025          CP0     = 10000.

      c        ireat   = iturb
      c        ISTART  = IREAT


0026          DO 101 L = iturb,  n-nsep, 1
0027            IF(CP(L).LT.CP0) THEN
0028              CP0     = CP(L)
0029              I0      = L
0030            ENDIF
0031    101   CONTINUE
0032          write(3,*) ' cp0,i0 , iturb, nsep n', cp0, i0 ,iturb, nsep,n

      c
      c         The separation point calculations are done on the upper
      c         surface begining at the equivalent turbulent
      c         boundary layer stagnation point and proceeding to the
      c         trailing edge. The counter in the inner do loop refers
      c         to the control point number.  All separation points
      c         are taken to be the upstream end point of the panel
      c         where separation has been determined.  (In other words,
      c         the leading edge of the separation panel.)
      c         Therefore:
      c
      c                 Xtsep(upper) = X(ITSEP(1))
```

```
        c
        c
        c
        c               Now this delta two is used to calculate the
        c               equivalent flat plate length (S0) for the
        c               Stratford method. (see Ducan, Mechanics of
        c               Fluids p.330)
        c
0033            C4          = 0.037       ! Ducan p.331
0034            S0          = (RE*((DELTA2/C4)**5)/C)**0.25
0035            write(3,*) ' s0, delta2 = ', s0, delta2
        c
        c               Next some values are initialized.
        c
0036            S           = S0
0037            CPMIN       = CP(I0)
0038            FNC1        = 0.0
0039            write(3,*) ' 102 i =iturb,n+1-nsep, ', iturb,n+1-nsep
        c         note this should be to n-nsep
0040            DO 102 I=iturb, n+1 - nsep, 1
0041              CP01    = (CP(I-1)-CPMIN)/(1.-CPMIN)
0042              CP02    = (CP(I)  -CPMIN)/(1.-CPMIN)
0043              CP03    = (CP(I+1)-CPMIN)/(1.-CPMIN)
        c
        c               Next the distances between the control points
        c               are found, where:
        c                       DS1     = distance between the i-1 and the ith
        c                                       control points
        c                       DS2     = distance between the i and the i+1th
        c                                       control points
        c
0044              DS1     = (SP(I-1)+SP(I))/2.
0045              DS2     = (SP(I)+SP(I+1))/2.
0046              DCP0DS  = -1.*CP01*DS2/(DS1*(DS1+DS2))
                +               + CP02*(1./DS1-1./DS2)
                +               + CP03*DS1/(DS2*(DS1+DS2))
0047              IF(I.EQ.iturb) THEN
0048                DCP0DS = (CP03-CP02)/DS2
0049              ENDIF
0050              D2CP    = CP(I-1)*(2./(DS1*(DS1+DS2)))
                +         - CP(I)*(2.*(DS1*DS2))
                +         + CP(I+1)*(2./(DS2*(DS1+DS2)))
0051              S       = S + DS1
0052              R       = RE*SQRT(1.-CPMIN)*(S/C)
        c
```

```
PDP-11 FORTRAN-77 V5.0-0        15:07:43    13-Nov-87          Page 100
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

        c                       Stratford's constants are set depending on
        c                       d2dp/dx2.
        c
0053            IF((D2CP.GE.0).AND.(CP02.LE.4./7.)) THEN
0054              B = 0.39
0055            ELSE IF (D2CP.LT.0.0) THEN
0056              B = 0.35
0057            ELSE
0058              write(3,*) 'TROUBLE IN STRATFORD! I, CP ',i,cp02
        c         TYPE *,'HIT <RETURN> TO CONTINUE'
        c         ACCEPT *
0059            ENDIF

0060            FNC     = FNC1
0061            RRHS    = CP02*SQRT(abs(S*DCP0DS))
0062            RLHS    = B*(R*0.000001)**0.1
0063            FNC1    = RRHS - RLHS
0064          write(3,*) ' i, fncl,cp01,cp02 ',i,fncl,cp01,cp02
0065            SIGN    = FNC1*FNC
        c
        c                       Finally separation is dedected once
        c                       SIGN changes sign.
        c

0066            IF(SIGN.LT.0.0) THEN
0067              ITSEP = I
0068              GOTO 200
0069            ENDIF
0070    102   CONTINUE
        c      used only if loop finds no separation point
0071          itsep = n+1 - nsep

0072    200   CONTINUE
0073          write(3,*) ' itsep ', itsep

0074          RETURN
0075          END


PROGRAM SECTIONS

 Name        Size                 Attributes

$CODE1   002246   595            RW,I,CON,LCL
$PDATA   000234    78            RW,D,CON,LCL
$IDATA   000046    19            RW,D,CON,LCL
$VARS    000126    43            RW,D,CON,LCL
$TEMPS   000002     1            RW,D,CON,LCL


Total Space Allocated = 002700   736
```

```
PDP-11 FORTRAN-77 V5.0-0        15:07:54    13-Nov-87          Page 101
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

        c
        c       Notation:
        c               s       = distance along the airfoil surface
        c                               from the stagnation point
        c               ds      = distance between control points on the
        c                               surface
        c               stsep   = distane along the surface to the turbulent
        c                               SEPARATION point
        c               0       = minium pressure reference
        c               CP0     = Stratford's conical Cp
        c                               p-p0
        c                       =     --------
        c                           0.5*rho*U0**2
        c               CP01    = CP0(i-1)
        c               CP02    = CP0(i)
        c               CP03    = CP0(i+1)
        c               DELTA2  = momentum thickness
        c
        c               NSURF= 1, ISURF = +1, & ILSEP(1)  => upper
        c               NSURF= 2, ISURF = -1, & ILSEP(2)  => lower
        c
0001            subroutine lcpout(ipi,ul,vl)

0002            include 'lfly.inc'
     *  c      file [20,64]lfly.inc
0003 *          integer    pmax
0004 *          parameter (pmax = 90) ! max number of panels
0005 *          integer    nwmax
0006 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0007 *          integer    nsmax
0008 *          parameter (nsmax=2)  ! max number of spanwise stations
0009 *          integer    ncmax
0010 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0011 *          integer    itmax
0012 *          parameter (itmax=5) ! max number of total iterations
0013            include 'lwingl.com'
     *  c      file [20,64]lwingl.com
     *
0014 *          integer  nw,                ! number of surfaces
     *          1                           ! wing + wake(s), & tunnel
     *          2        ign,               ! number of gammas
     *          3        ipn                ! number of panels
0015 *          real     la1(nwmax),
     *          1        la2(nwmax),
     *          2        sw(nwmax),         ! half span = b/2
     *          2        cwak(nwmax),       ! length of wake
     *          3        cw(nwmax),         ! chord of wing
     *          4        x0(nwmax),
     *          5        z0(nwmax),
     *          6        alpha(nwmax),
     *          7        dihed(nwmax),
     *          8        eh,                ! tunnel ellipse height
     *          9        gbh,               ! tunnel ground board height
     *          1        reyn               ! reynolds number e-6
```

```
PDP-11 FORTRAN-77 V5.0-0        15:07:54    13-Nov-87          Page 102
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0016 *          integer  stype(nwmax)      ! surface type wing wake or tunnel
     *  c                                    >0 NACA 4 digit airfoil
     *  c                                    =0 NACA 64A005
     *  c                                    = -10 tunnel
     *  c
0017 *          integer  nc(nwmax),        ! wing chordwise panels
     *          2        nct(nwmax),       ! total chordwise panels
     *          2        ns(nwmax),        ! spanwise panels
     *          3       !bc(nwmax),        ! boundary condition
     *          4        nsep(nwmax,nsmax)! separation pt
     *  c                separates at nsep panels upstream of trailing edge
0018 *          integer  itws,             ! wake shape iterations
     *          6        itsp,             ! separation pt iterations
     *          7        itto              ! total iterations
     *
     *
0019 *          common /lwingl/ nw, ign, ipn, la1, la2, sw, cwak, cw, x0, z0,
     *          1               alpha, dihed, eh, gbh, reyn, stype,
     *          2               nc, nct, ns, nsep, itws, itsp, itto
     *
0020            include 'lchar.com'
     *  c      file [20,64]lchar.com
     *
0021 *          character*40  title
0022 *          character*4  run
     *
0023 *          common /lchar/  title,run
     *

0024            character*10 fname
0025            integer ipi
0026            virtual ipi(nwmax,2,nsmax,ncmax)
0027            real    ul(pmax),vl(pmax)

0028            pi = acos(-1.0)

0029            fname(1:10) = 'lcpxxx.xxx'
0030            fname(8:10) = run(1:3)

0031            do 101 inw = 1, nw
0032              write (fname(4:4), '(il)') inw
0033              do 111 ins = 1, ns(inw)
0034                write (fname(5:5), '(il)') ins
0035                do 121 iull = 1,2
0036                write (fname(6:6), '(il)') iull
0037                  open (unit=1,name=fname,status='new')
0038                  do 131 incl = 1,nc(inw)

       c                xple = (0.5-0.5*cos((incl-1.)/nc(inw)*pi))
       c                xpte = (0.5-0.5*cos((incl*1.)/nc(inw)*pi))

0039              xple  = (1.0-    cos((incl-1.)/nc(inw)*pi/2))
0040              xpte  = (1.0-    cos((incl*1.)/nc(inw)*pi/2))

0041                    ipl = ipi(inw,iull,ins,incl)
```

```
PDP-11 FORTRAN-77 V5.0-0        15:07:54    13-Nov-87          Page 103
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0042                    write(1,*) (xple+xpte)*100./2.,
               1        - (1.0 - ul(ipl)**2 - vl(ipl)**2 )

0043     131          continue
0044                  close(1)
0045     121        continue
0046     111      continue
0047     101    continue
0048            end


PROGRAM SECTIONS

 Name        Size                  Attributes

$CODE1   001164   314              RW,I,CON,LCL
$PDATA   000072    29              RW,D,CON,LCL
$IDATA   000036    15              RW,D,CON,LCL
$VARS    000040    16              RW,D,CON,LCL
$TEMPS   000020     8              RW,D,CON,LCL
LWING1   000170    60              RW,D,OVR,GBL
LCHAR    000054    22              RW,D,OVR,GBL


Total Space Allocated = 001640   464
```

```
0001            subroutine lshout(ipi,
                1                 iga, iul, inc, iwng,
                2                 la, theta, chi, c, b,
                3                 xp, yp, zp, xc, yc, zc )

0002            include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0003 *          integer    pmax
0004 *          parameter (pmax = 90) ! max number of panels
0005 *          integer    nwmax
0006 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0007 *          integer    nsmax
0008 *          parameter (nsmax=2)  ! max number of spanwise stations
0009 *          integer    ncmax
0010 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0011 *          integer    itmax
0012 *          parameter (itmax=5) ! max number of total iterations
0013            include 'lwing1.com'
     *  c       file [20,64]lwing1.com
     *
0014 *          integer  nw,                ! number of surfaces
     *          1                           ! wing + wake(s), & tunnel
     *          2        ign,               ! number of gammas
     *          3        ipn                ! number of panels
0015 *          real     la1(nwmax),
     *          1        la2(nwmax),
     *          2        sw(nwmax),         ! half span = b/2
     *          2        cwak(nwmax),       ! length of wake
     *          3        cw(nwmax),         ! chord of wing
     *          4        x0(nwmax),
     *          5        z0(nwmax),
     *          6        alpha(nwmax),
     *          7        dihed(nwmax),
     *          8        eh,                ! tunnel ellipse height
     *          9        gbh,               ! tunnel ground board height
     *          1        reyn               ! reynolds number e-6
0016 *          integer  stype(nwmax)       ! surface type wing wake or tunnel
     *  c                                     >0 NACA 4 digit airfoil
     *  c                                     =0 NACA 64A005
     *  c                                     = -10 tunnel
     *  c
0017 *          integer  nc(nwmax),         ! wing chordwise panels
     *          2        nct(nwmax),        ! total chordwise panels
     *          2        ns(nwmax),         ! spanwise panels
     *          3       !bc(nwmax),         ! boundary condition
     *          4        nsep(nwmax,nsmax)! separation pt
     *  c                separates at nsep panels upstream of trailing edge
0018 *          integer  itws,              ! wake shape iterations
     *          6        itsp,              ! separation pt iterations
     *          7        itto               ! total iterations
     *
     *
0019 *          common /lwing1/ nw, ign, ipn, la1, la2, sw, cwak, cw, x0, z0,
```

```
PDP-11 FORTRAN-77 V5.0-0        15:08:01    13-Nov-87          Page 105
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

     *          1               alpha, dihed, eh, gbh, reyn, stype,
     *          2               nc, nct, ns, nsep, itws, itsp, itto
     *
0020            include 'lwing2.var'
     *  c       file 'lwing2.var'
     *  c       changed from common block to allow use of virtual memory
     *
0021 *          integer iga,
     *          1       iul,
     *          2       inc,
     *          3       iwng
0022 *          real    la,
     *          1       theta,
     *          2       chi,
     *          3       c,
     *          4       b,
     *          5       xp, yp, zp,
     *          6       xc, yc, zc
     *
0023 *         virtual    iga(pmax), iul(pmax), inc(pmax), iwng(pmax),
     *       1            la(pmax), theta(pmax), chi(pmax), c(pmax), b(pmax),
     *       2            xp(pmax), yp(pmax), zp(pmax),
     *       3            xc(pmax), yc(pmax), zc(pmax)
     *
0024            include 'lchar.com'
     *  c       file [20,64]lchar.com
     *
0025 *          character*40  title
0026 *          character*4  run
     *
0027 *          common /lchar/  title,run
     *

0028            character*11 fname
0029            integer ipi
0030            virtual ipi(nwmax,2,nsmax,ncmax)

0031            pi = acos(-1.0)

0032            fname(1:11) = 'lshxxxx.xxx'
0033            fname(9:11) = run(1:3)

0034            do 101 inw = 1, nw
0035              write (fname(4:4), '(i1)') inw
0036              do 111 ins = 1, ns(inw)
0037                write (fname(5:5), '(i1)') ins
0038                write (fname(6:7), '(i2)') itto
0039                open (unit=1,name=fname,status='new')

        c           write x,z position of the trailing edge upper surface

0040                iull = 1
0041                ipl = ipi(inw,iull,ins,nct(inw))
0042                write (1,*) xp(ipl)+c(ipl)*cos(theta(ipl)),
                1               zp(ipl)+c(ipl)*sin(theta(ipl))
```

```
      c           write upper surface to leading edge
0043              do 121 incl = nct(inw), 1, -1
0044                ipl = ipi(inw,iull,ins,incl)
0045                write (1,*) xp(ipl), zp(ipl)
0046  121         continue

      c           lower surface
0047              iull = 2
0048              do 122 incl = 2, nct(inw)
0049                ipl = ipi(inw,iull,ins,incl)
0050                write (1,*) xp(ipl), zp(ipl)
0051  122         continue

0052              ipl = ipi(inw,iull,ins,nct(inw))
0053              write (1,*) xp(ipl)+c(ipl)*cos(theta(ipl)),
     1                        zp(ipl)+c(ipl)*sin(theta(ipl))

0054              close(1)
0055  111       continue
0056  101     continue
0057            end
```

PROGRAM SECTIONS

Name	Size		Attributes
$CODE1	001676	479	RW,I,CON,LCL
$PDATA	000074	30	RW,D,CON,LCL
$IDATA	000240	80	RW,D,CON,LCL
$VARS	000032	13	RW,D,CON,LCL
$TEMPS	000022	9	RW,D,CON,LCL
LWING1	000170	60	RW,D,OVR,GBL
LCHAR	000054	22	RW,D,OVR,GBL

Total Space Allocated = 002552 693

```
PDP-11 FORTRAN-77 V5.0-0        15:08:11    13-Nov-87          Page 107
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR


0001          subroutine lclout

0002          include 'lfly.inc'
     *  c       file [20,64]lfly.inc
0003 *          integer    pmax
0004 *          parameter (pmax = 90) ! max number of panels
0005 *          integer    nwmax
0006 *          parameter (nwmax=2)  ! max number of surfaces (wings)
0007 *          integer    nsmax
0008 *          parameter (nsmax=2)  ! max number of spanwise stations
0009 *          integer    ncmax
0010 *          parameter (ncmax=44)  ! max number of chordwise stations
     *  c                               on a wing including wake
0011 *          integer    itmax
0012 *          parameter (itmax=5) ! max number of total iterations
0013          include 'lwing1.com'
     *  c       file [20,64]lwing1.com
     *
0014 *          integer  nw,                ! number of surfaces
     *          1                           ! wing + wake(s), & tunnel
     *          2        ign,               ! number of gammas
     *          3        ipn                ! number of panels
0015 *          real     la1(nwmax),
     *          1        la2(nwmax),
     *          2        sw(nwmax),         ! half span = b/2
     *          2        cwak(nwmax),       ! length of wake
     *          3        cw(nwmax),         ! chord of wing
     *          4        x0(nwmax),
     *          5        z0(nwmax),
     *          6        alpha(nwmax),
     *          7        dihed(nwmax),
     *          8        eh,                ! tunnel ellipse height
     *          9        gbh,               ! tunnel ground board height
     *          1        reyn               ! reynolds number e-6
0016 *          integer  stype(nwmax)       ! surface type wing wake or tunnel
     *  c                                     >0 NACA 4 digit airfoil
     *  c                                     =0 NACA 64A005
     *  c                                     = -10 tunnel
     *  c
0017 *          integer  nc(nwmax),         ! wing chordwise panels
     *          2        nct(nwmax),        ! total chordwise panels
     *          2        ns(nwmax),         ! spanwise panels
     *          3       !bc(nwmax),         ! boundary condition
     *          4        nsep(nwmax,nsmax)! separation pt
     *  c                separates at nsep panels upstream of trailing edge
0018 *          integer  itws,              ! wake shape iterations
     *          6        itsp,              ! separation pt iterations
     *          7        itto               ! total iterations
     *
     *
0019 *          common /lwing1/ nw, ign, ipn, la1, la2, sw, cwak, cw, x0, z0,
     *          1               alpha, dihed, eh, gbh, reyn, stype,
     *          2               nc, nct, ns, nsep, itws, itsp, itto
     *
```

```
PDP-11 FORTRAN-77 V5.0-0        15:08:11    13-Nov-87          Page 108
LFLY.TMP;1              /F77/OP/TR:BLOCKS/WR

0020          include 'lchar.com'
     *  c       file [20,64]lchar.com
     *
0021 *          character*40  title
0022 *          character*4  run
     *
0023 *          common /lchar/  title,run
     *
0024          include 'lhist.com'
     *  c       file [20,64]lhist.com
     *  c       Used to keep information for convergence history of method
     *
     *
0025 *          real    cl(nwmax,itmax) !cl vs iteration history
     *
0026 *          common /lhist/ cl
     *
     *

0027          character*8 fname

0028          fname(1:8) = 'lclx.xxx'
0029          fname(6:8) = run(1:3)

0030          do 101 inw = 1, nw
0031            write (fname(4:4), '(il)') inw
0032            open (unit=1,name=fname,status='new')
0033            do 111 it = 1,itto
0034              write(1,*) it, cl(inw,it)
0035    111     continue
0036            close(1)
0037    101   continue
0038          end


PROGRAM SECTIONS

 Name        Size                 Attributes

$CODE1   000360   120             RW,I,CON,LCL
$PDATA   000050    20             RW,D,CON,LCL
$VARS    000014     6             RW,D,CON,LCL
$TEMPS   000002     1             RW,D,CON,LCL
LWING1   000170    60             RW,D,OVR,GBL
LCHAR    000054    22             RW,D,OVR,GBL
LHIST    000050    20             RW,D,OVR,GBL


Total Space Allocated = 000762   249

No FPP Instructions Generated
```

FIGURES

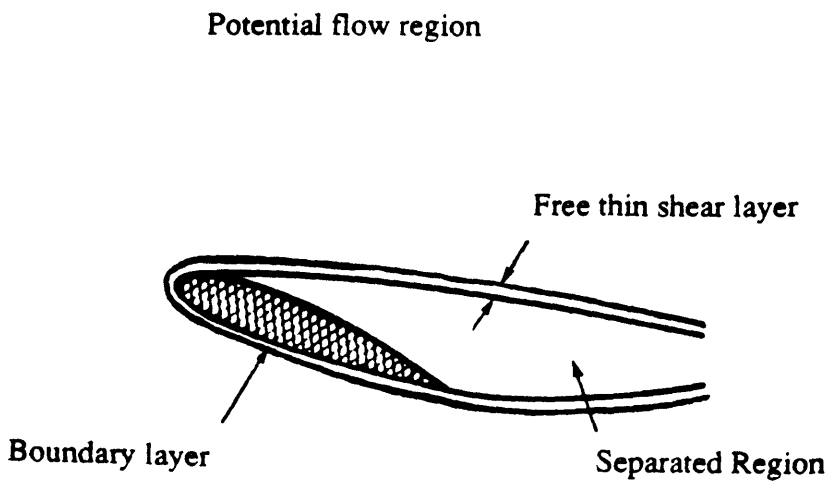


Figure 2-1 Flow regions surrounding an airfoil

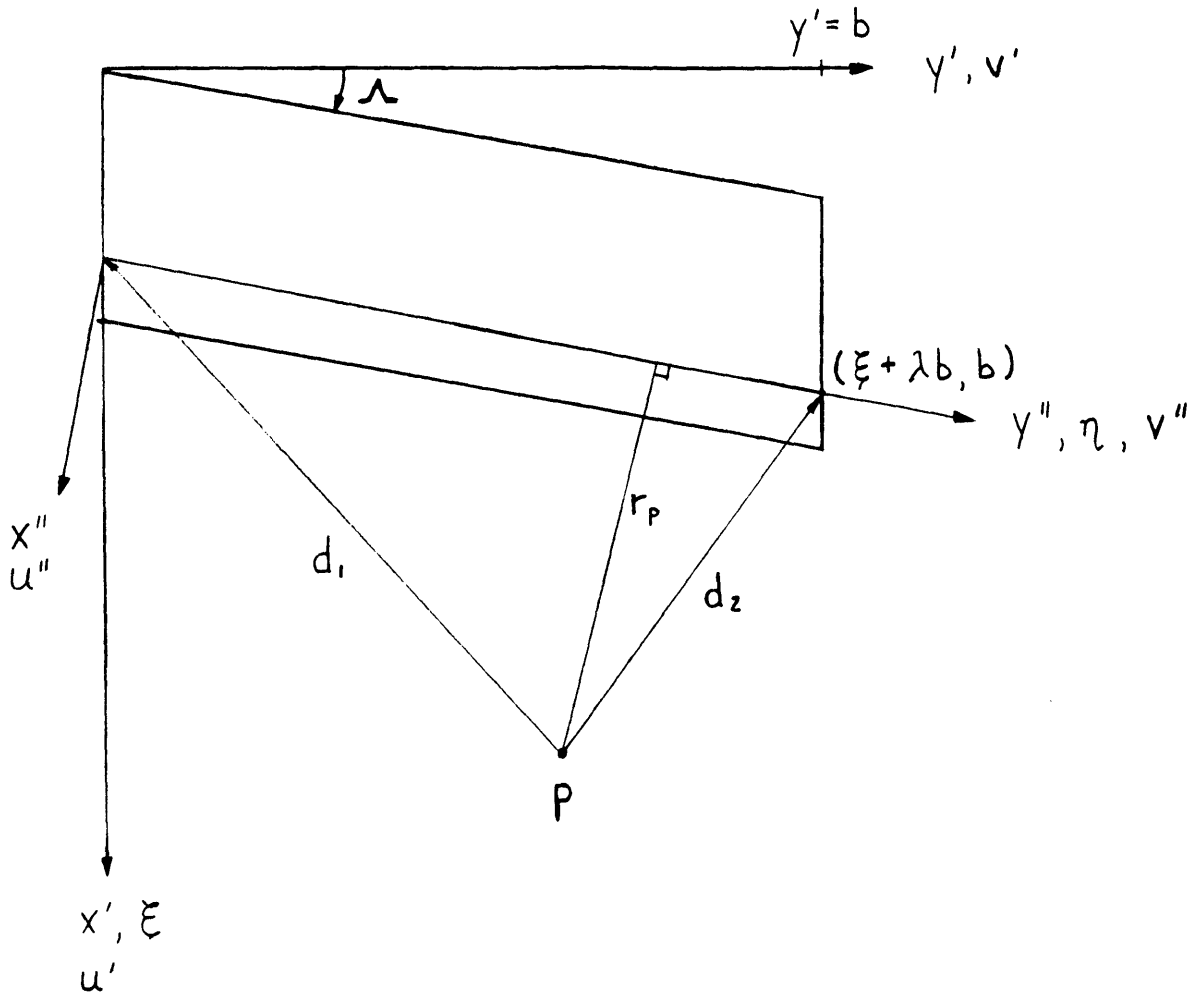


Figure 4-1 Panel planform

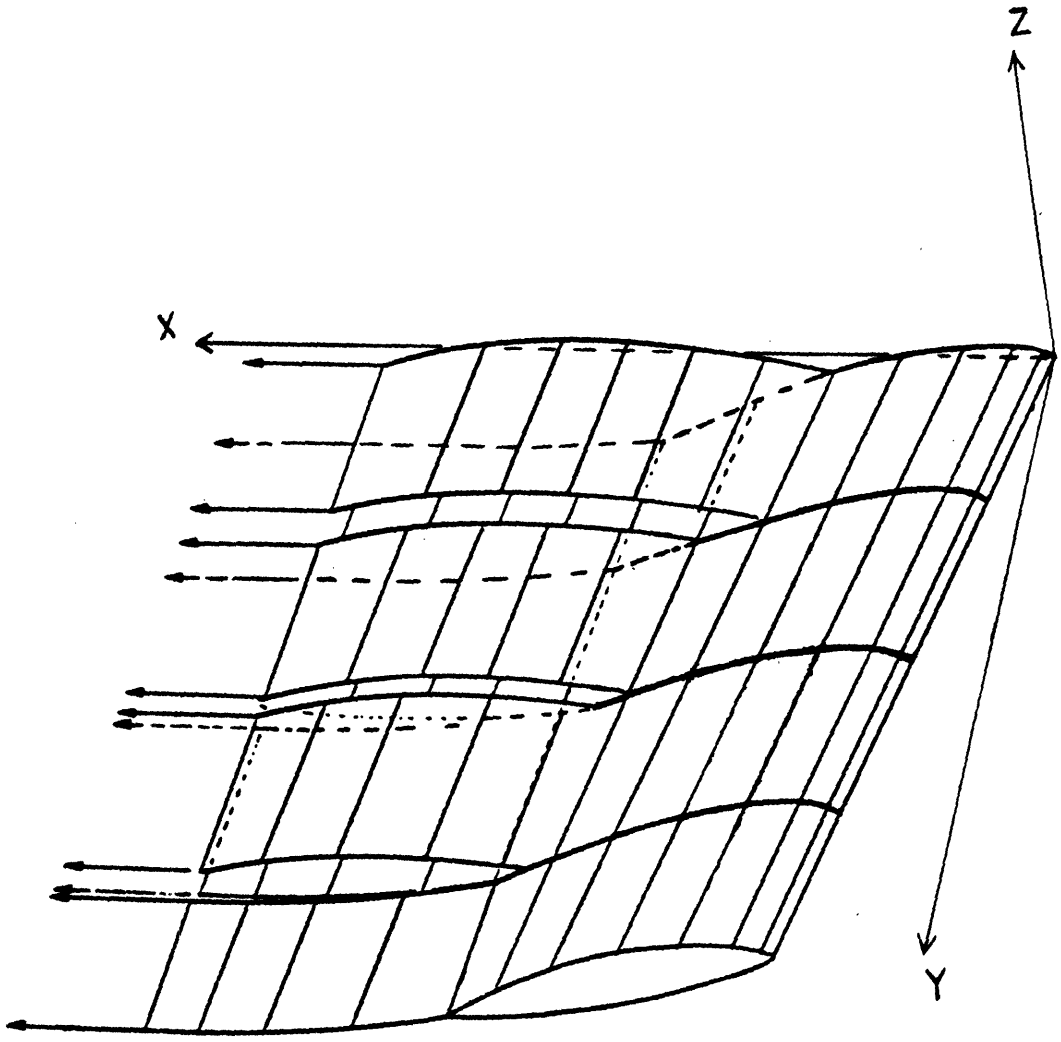


Figure 4-2 Wing paneling and coordinate system

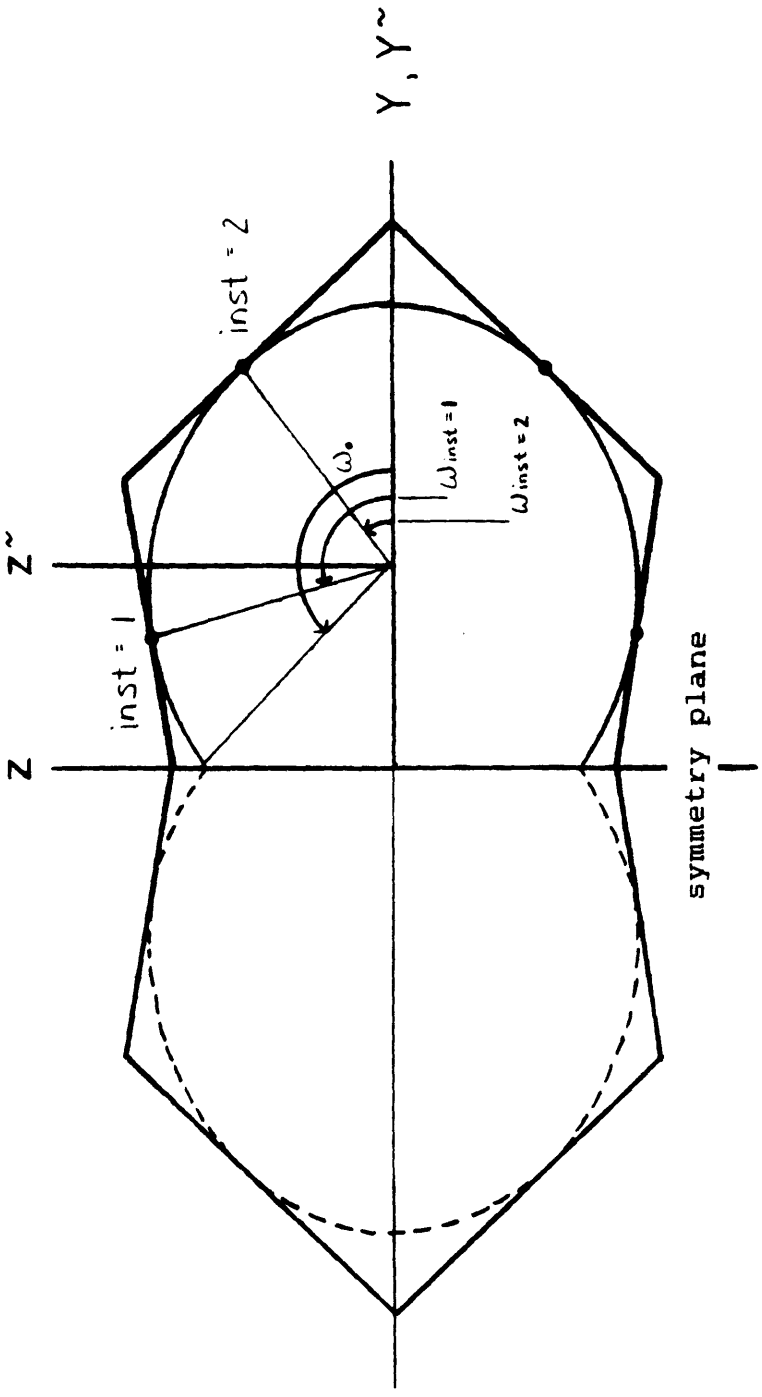


Figure 4-3 Tunnel cross section

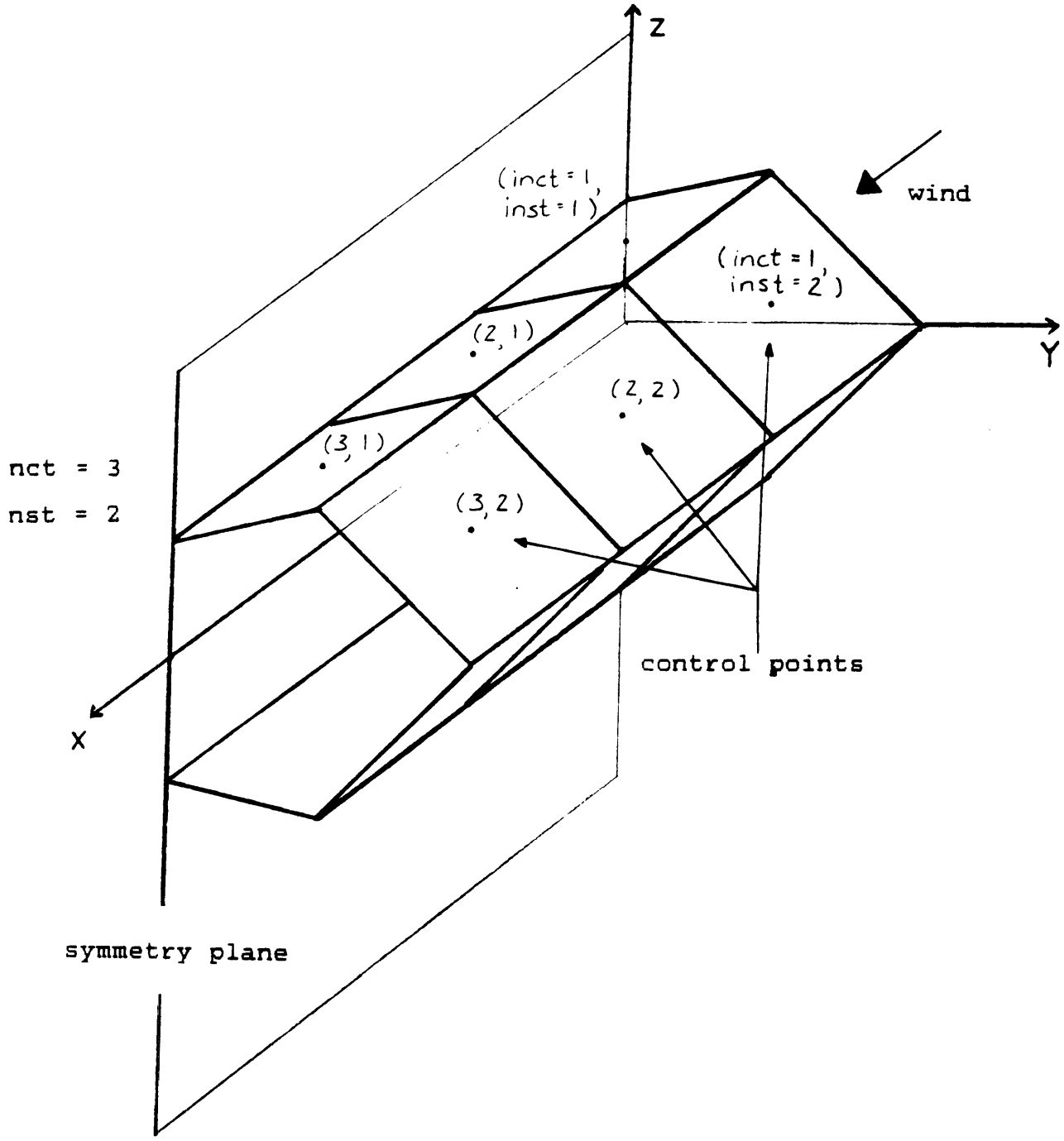


Figure 4-4 Tunnel nomenclature and coordinate system

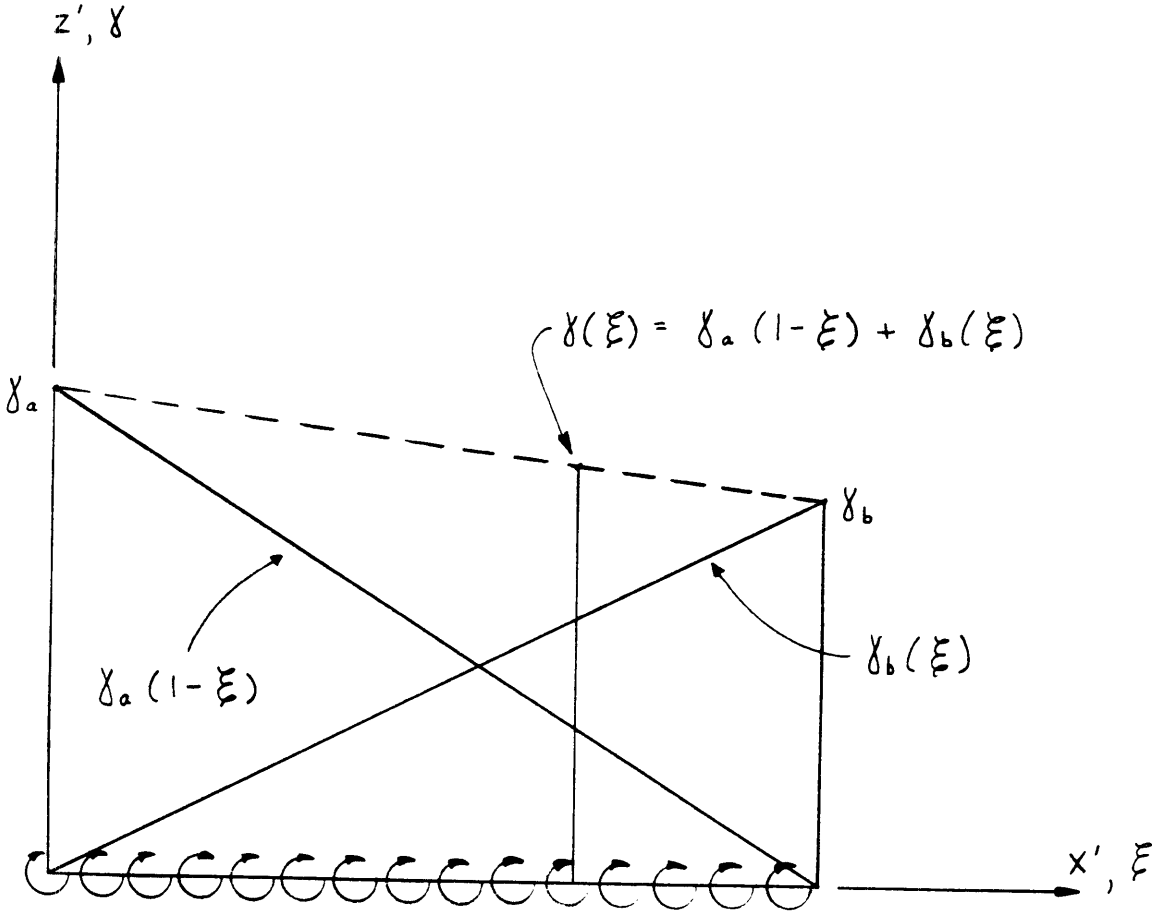


Figure 4-5 Panel vorticity distribution

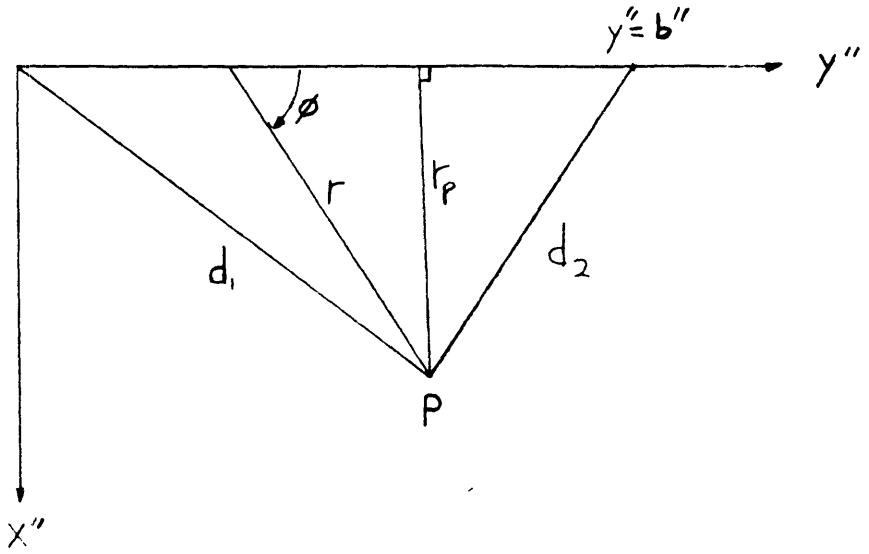


Figure 4-6 Vortex filament coordinate system

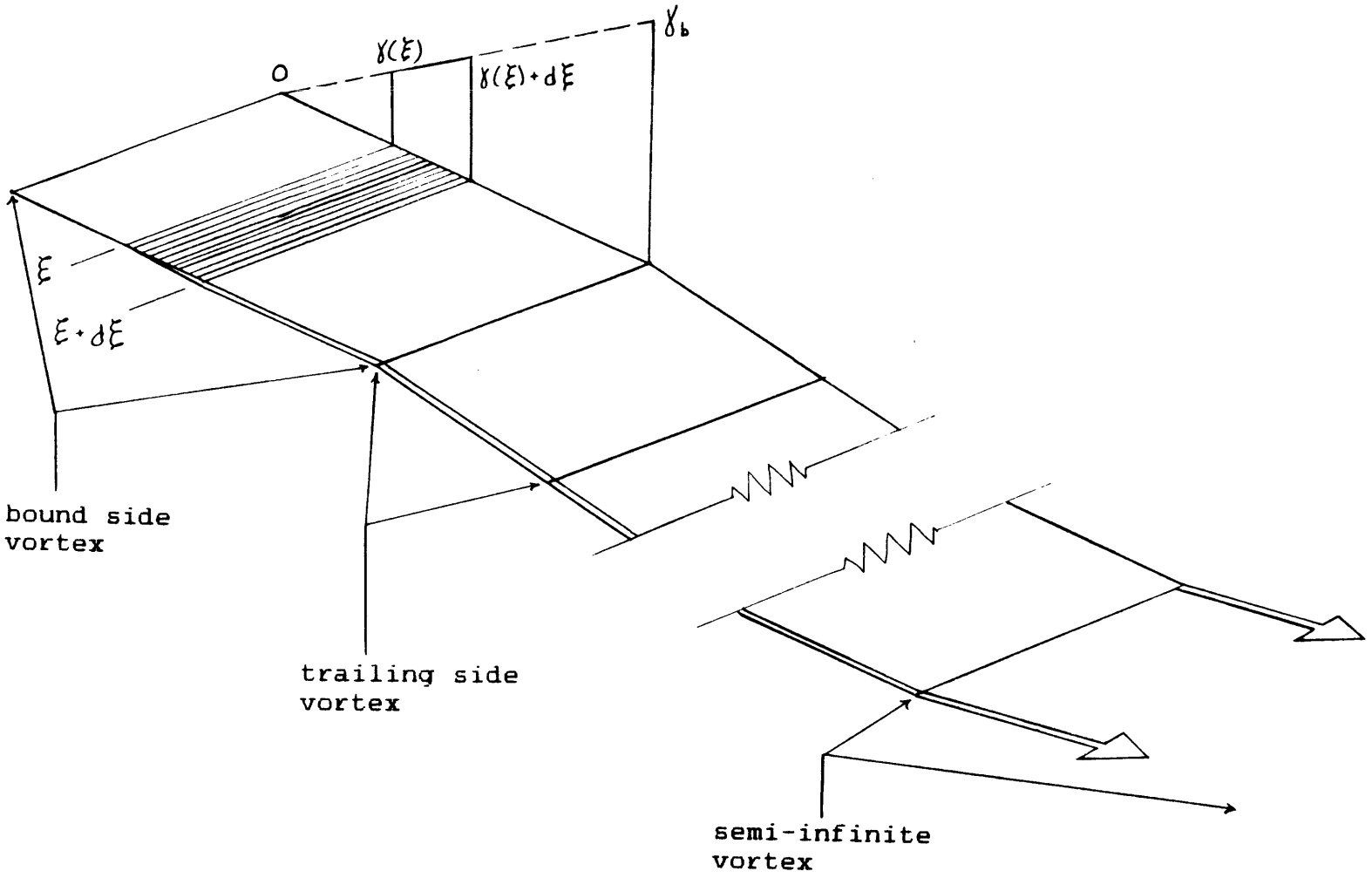


Figure 4-7 Vortex filament sections

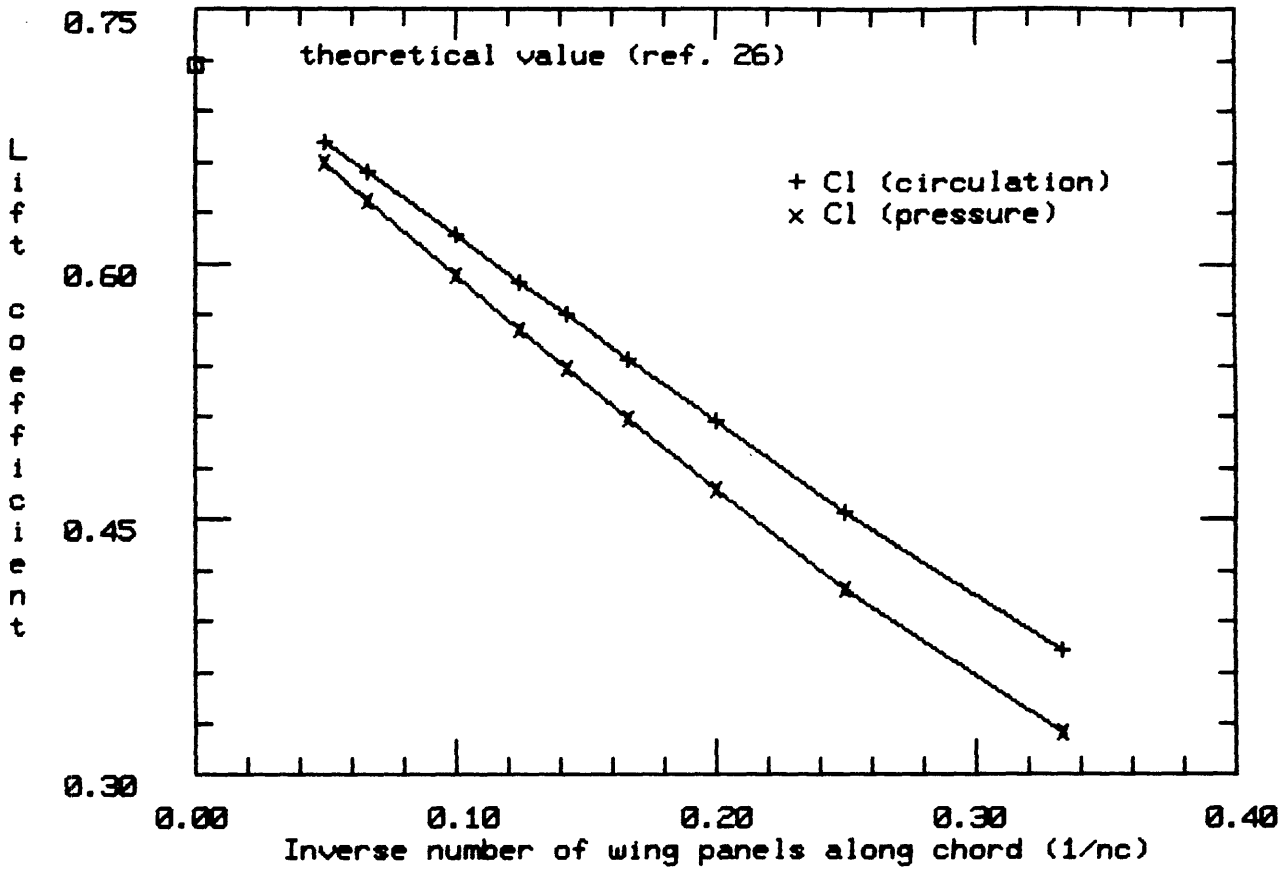


Figure 6-1 Lift versus inverse number of wing panels along chord

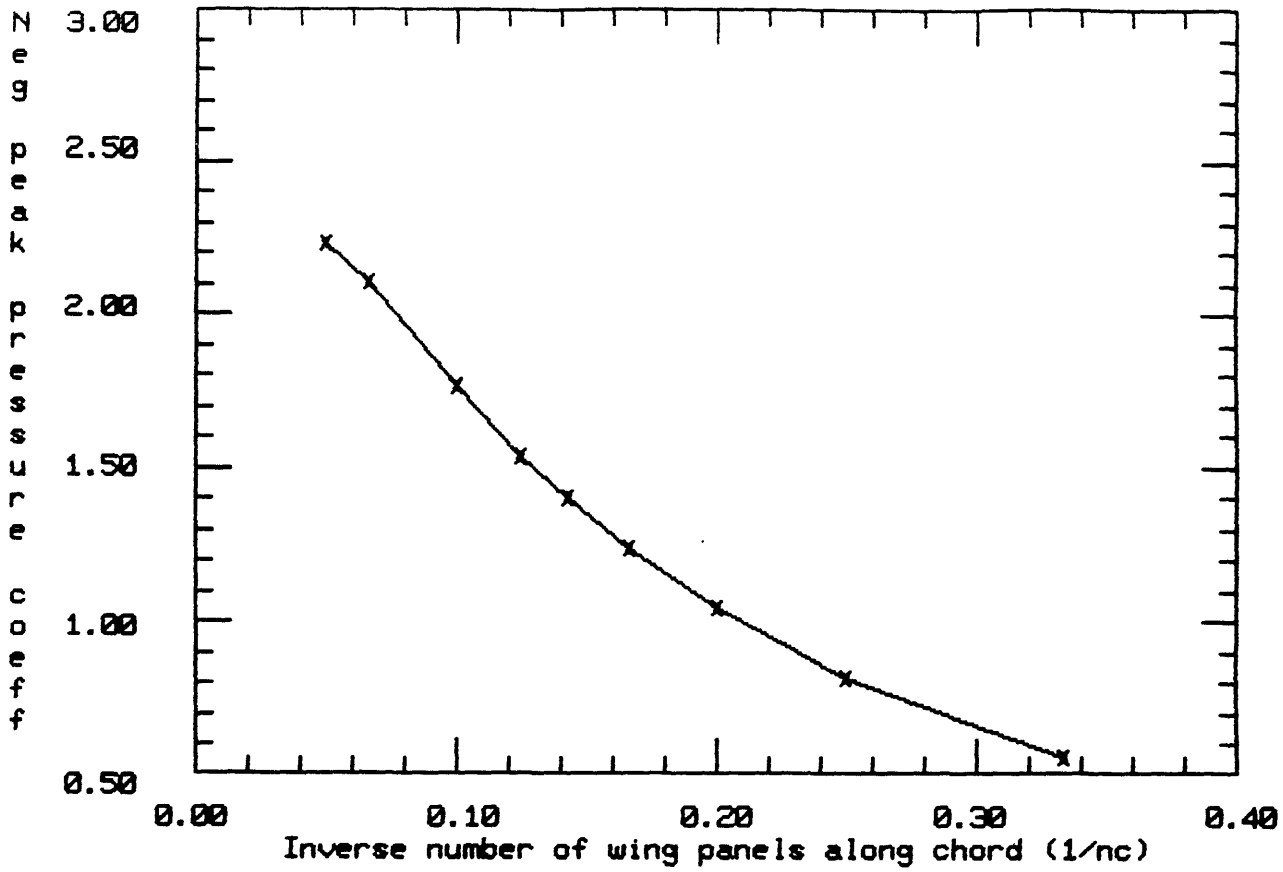


Figure 6-2 Negative peak pressure coefficient versus inverse number of wing panels along chord

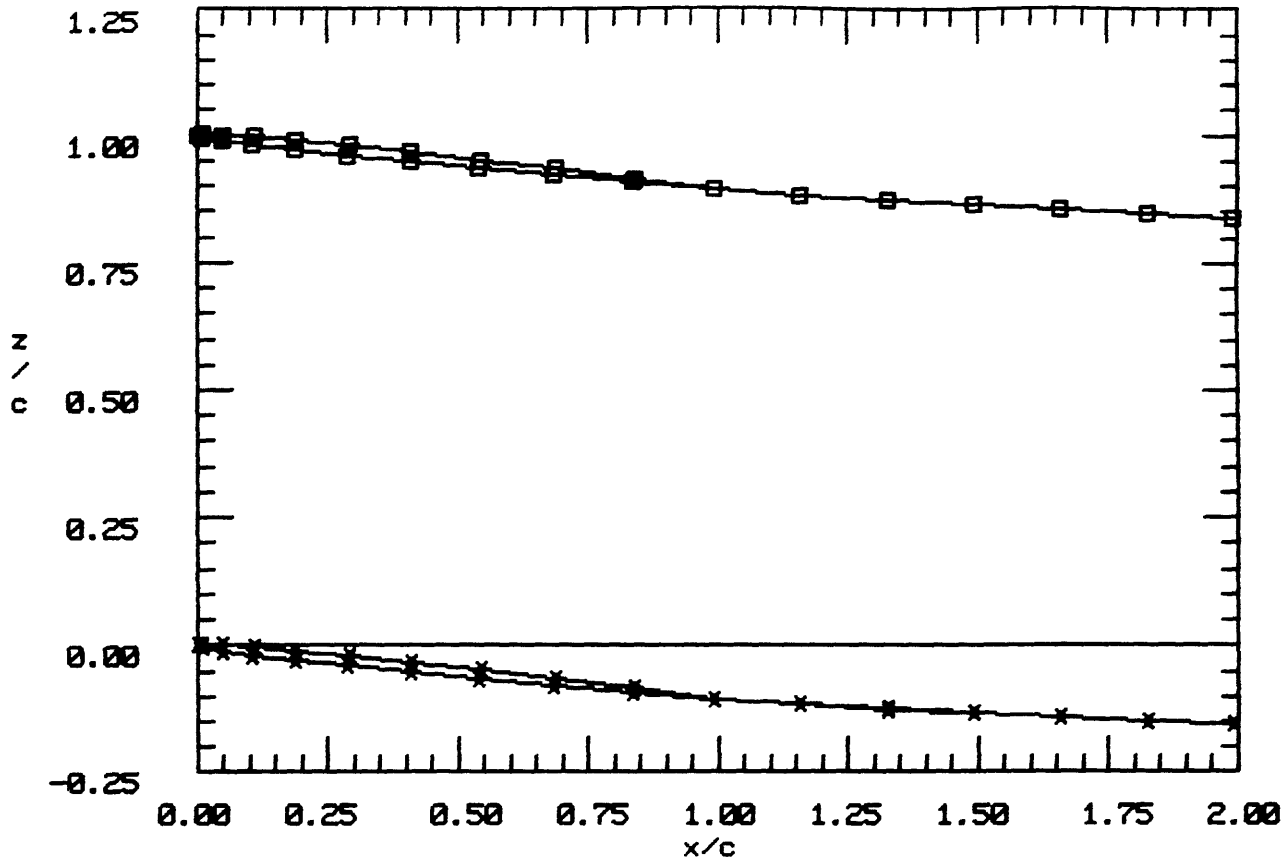


Figure 6-3 Thin airfoil 2-D biplane and wake geometry

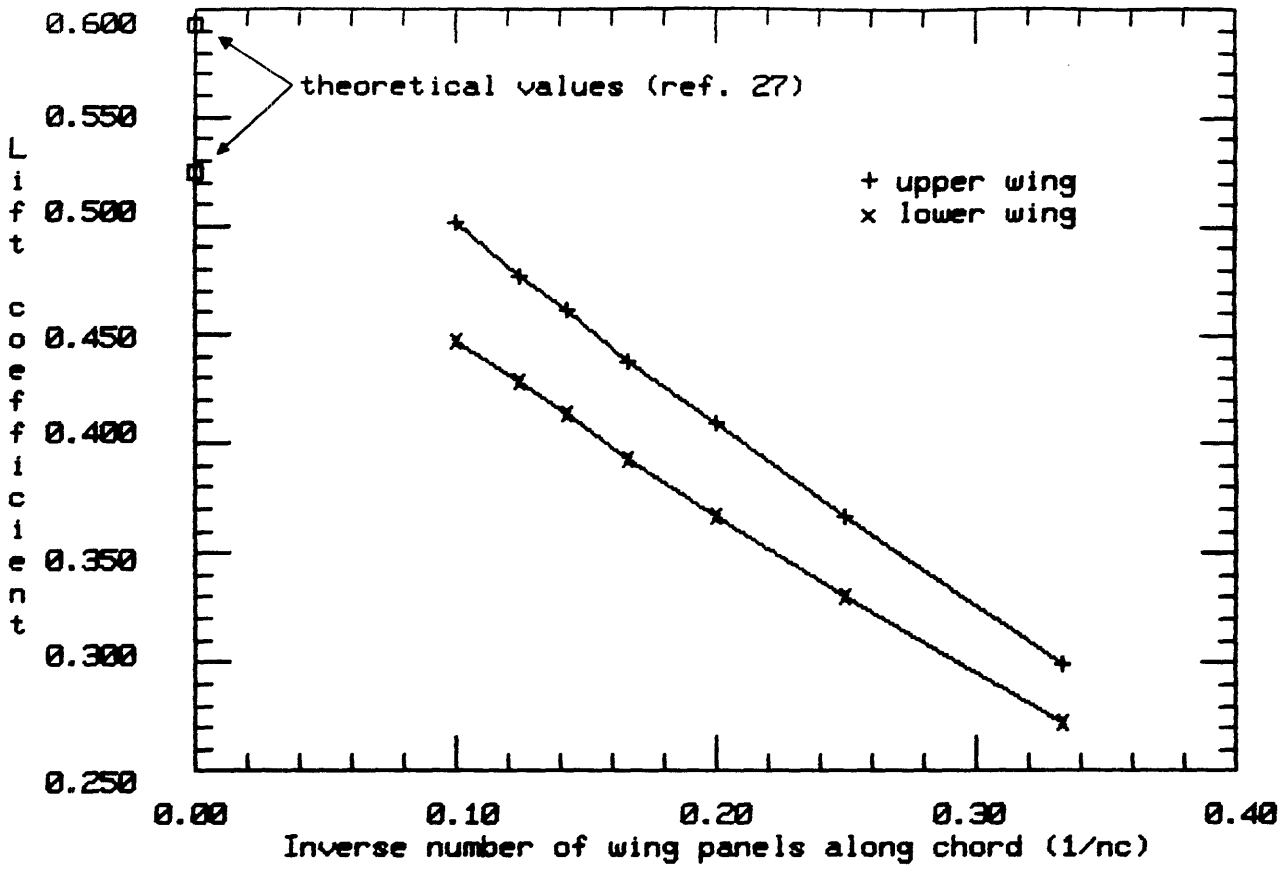


Figure 6-4 2-D biplane lift versus inverse number of wing panels along chord

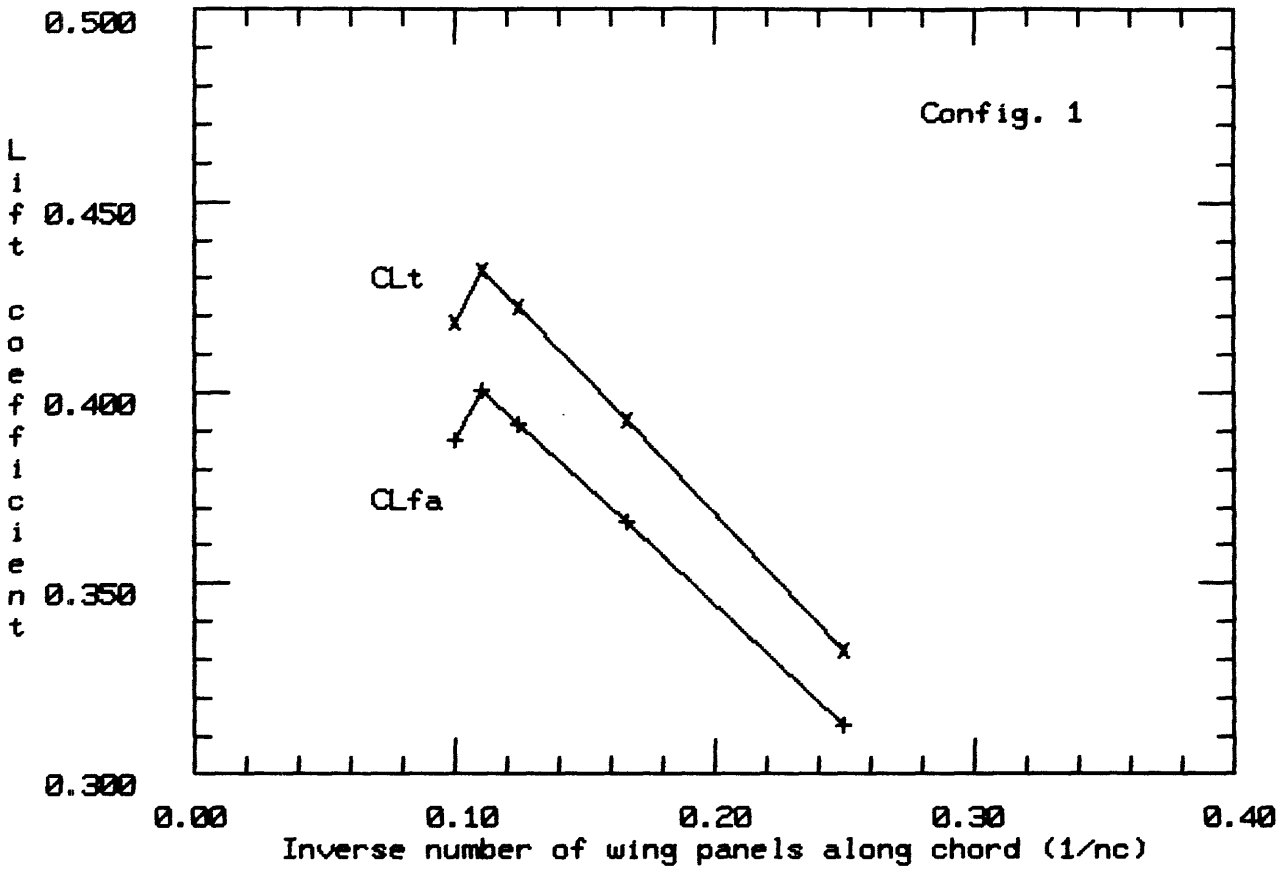


Figure 6-5 Lift coefficient versus inverse number of wing panels along chord

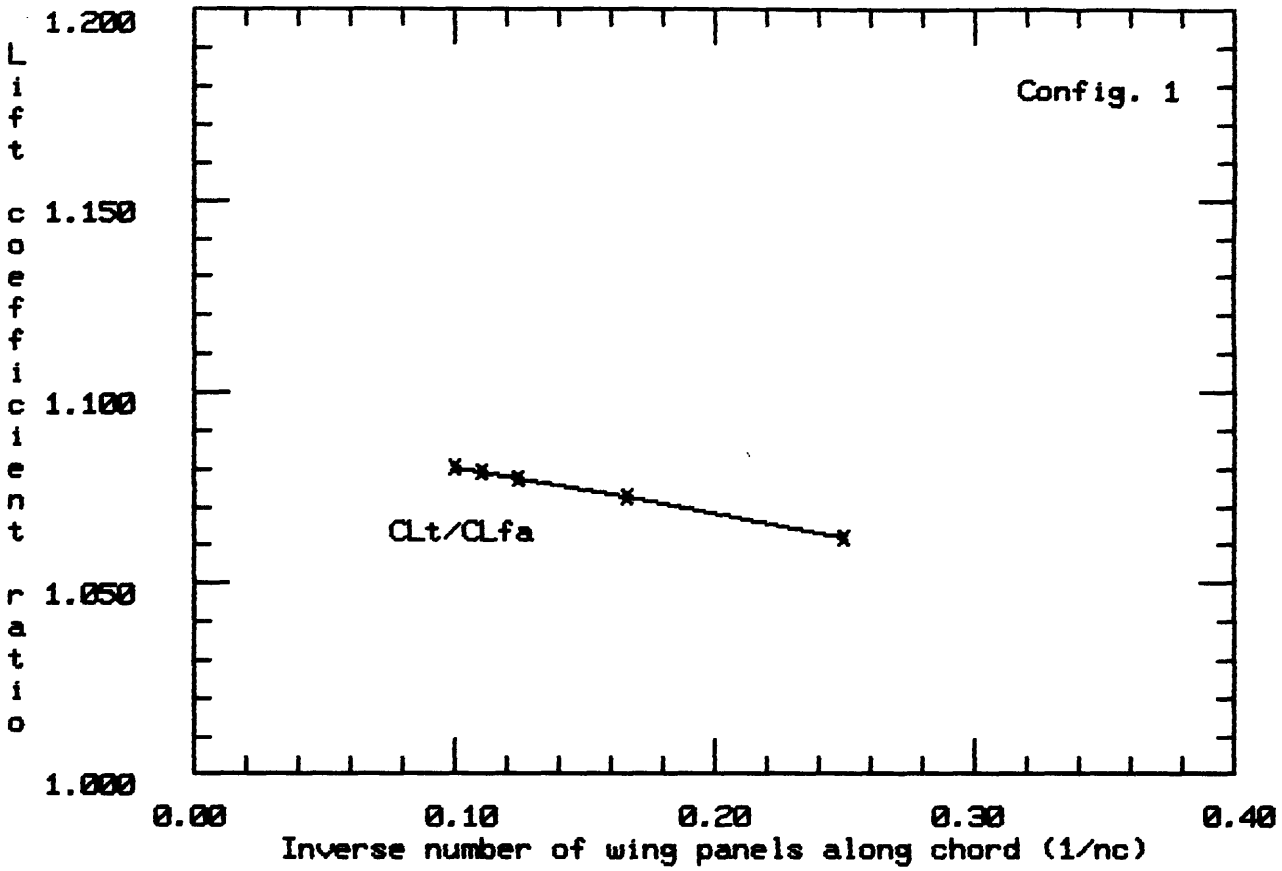


Figure 6-6 Lift coefficient ratio versus inverse number of wing panels along chord

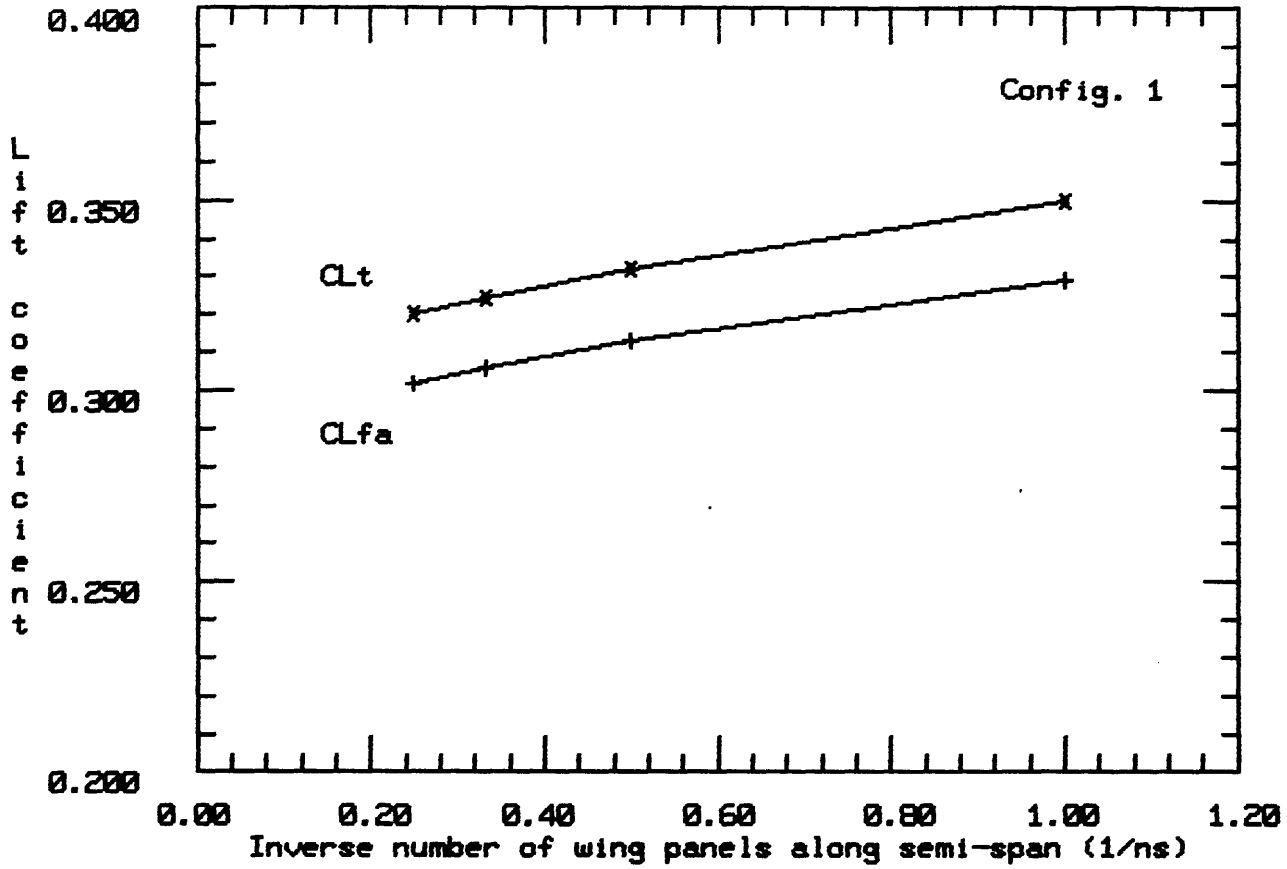


Figure 6-7 Lift coefficient versus inverse number of wing semi-span panels

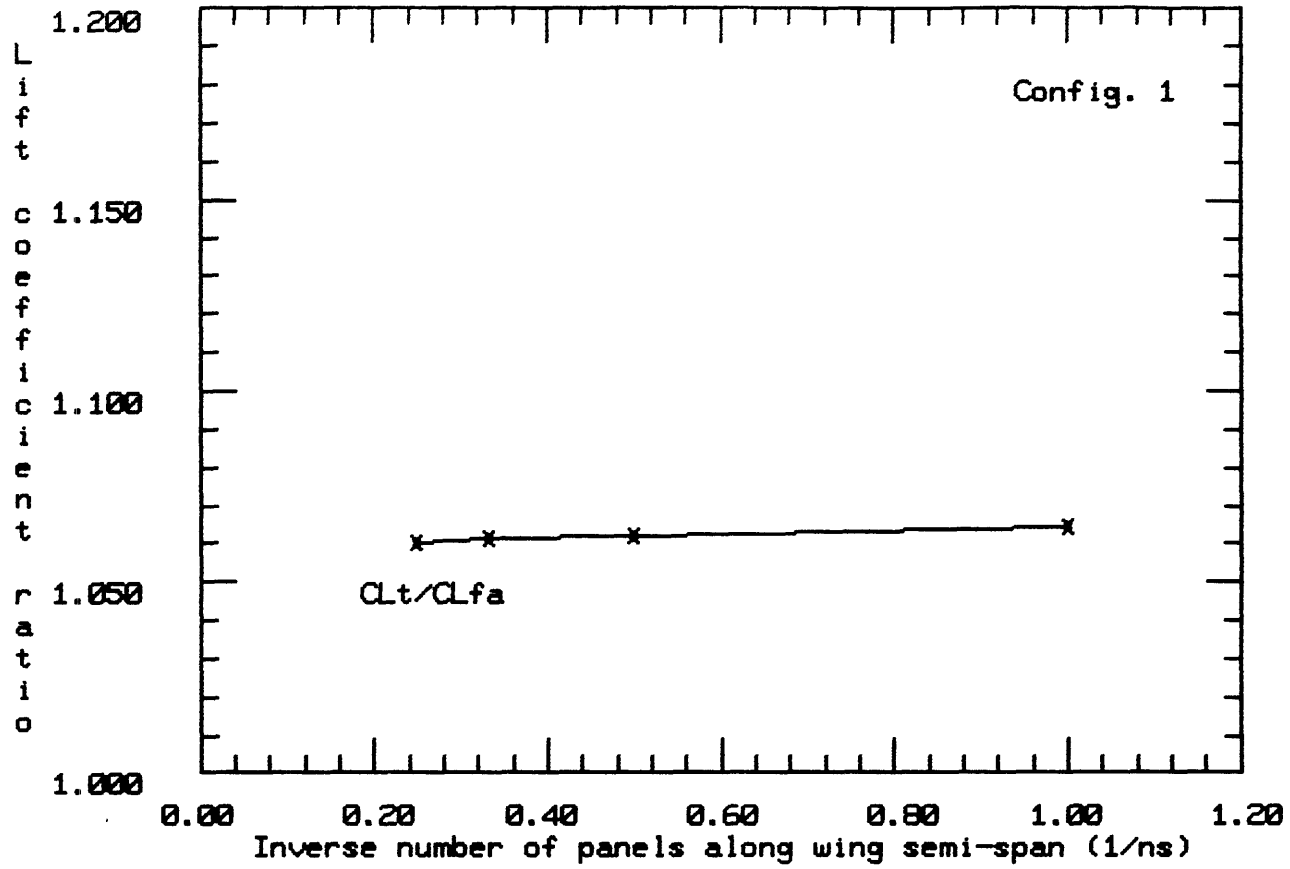


Figure 6-8 Lift coefficient ratio versus inverse number of wing semi-span panels

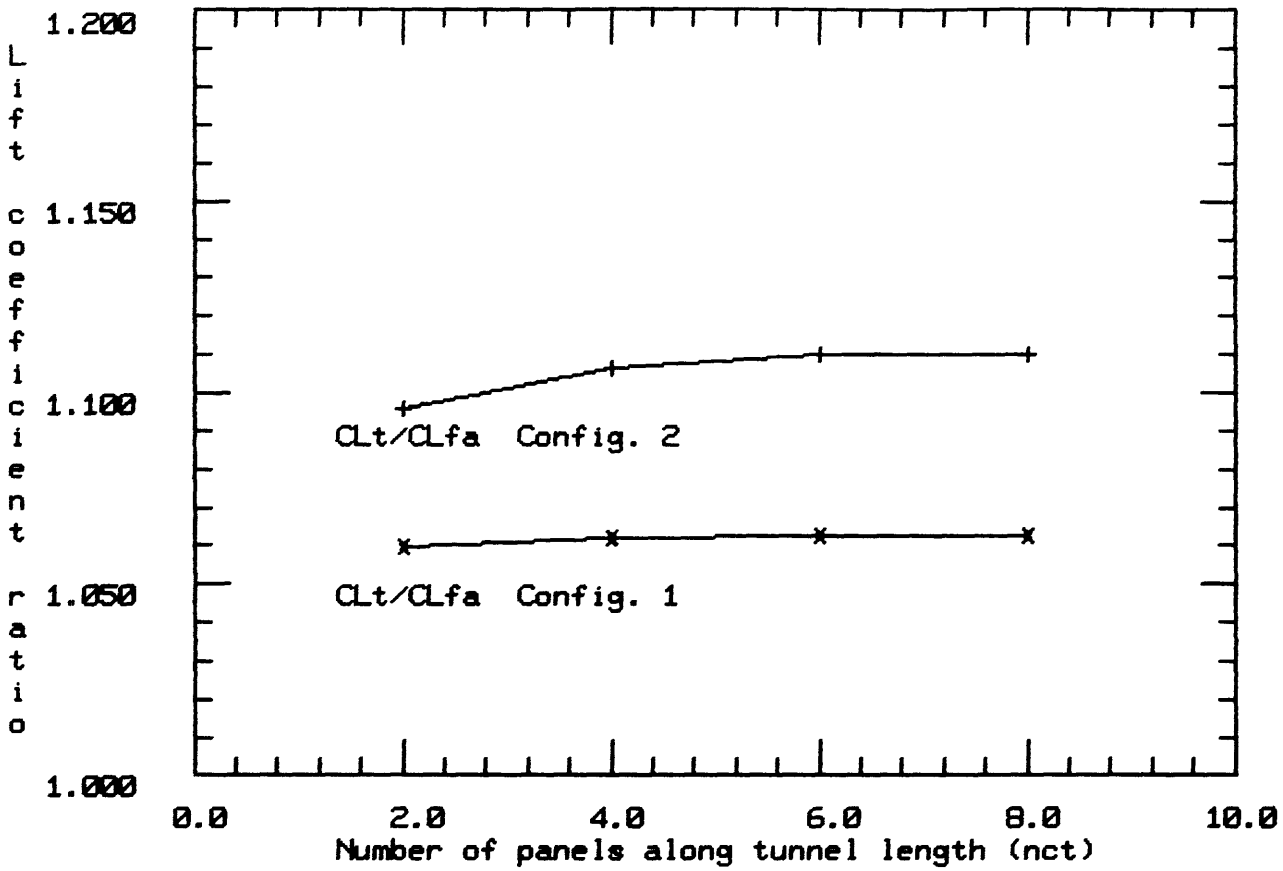


Figure 6-9 Lift coefficient ratio versus number of panels along tunnel length

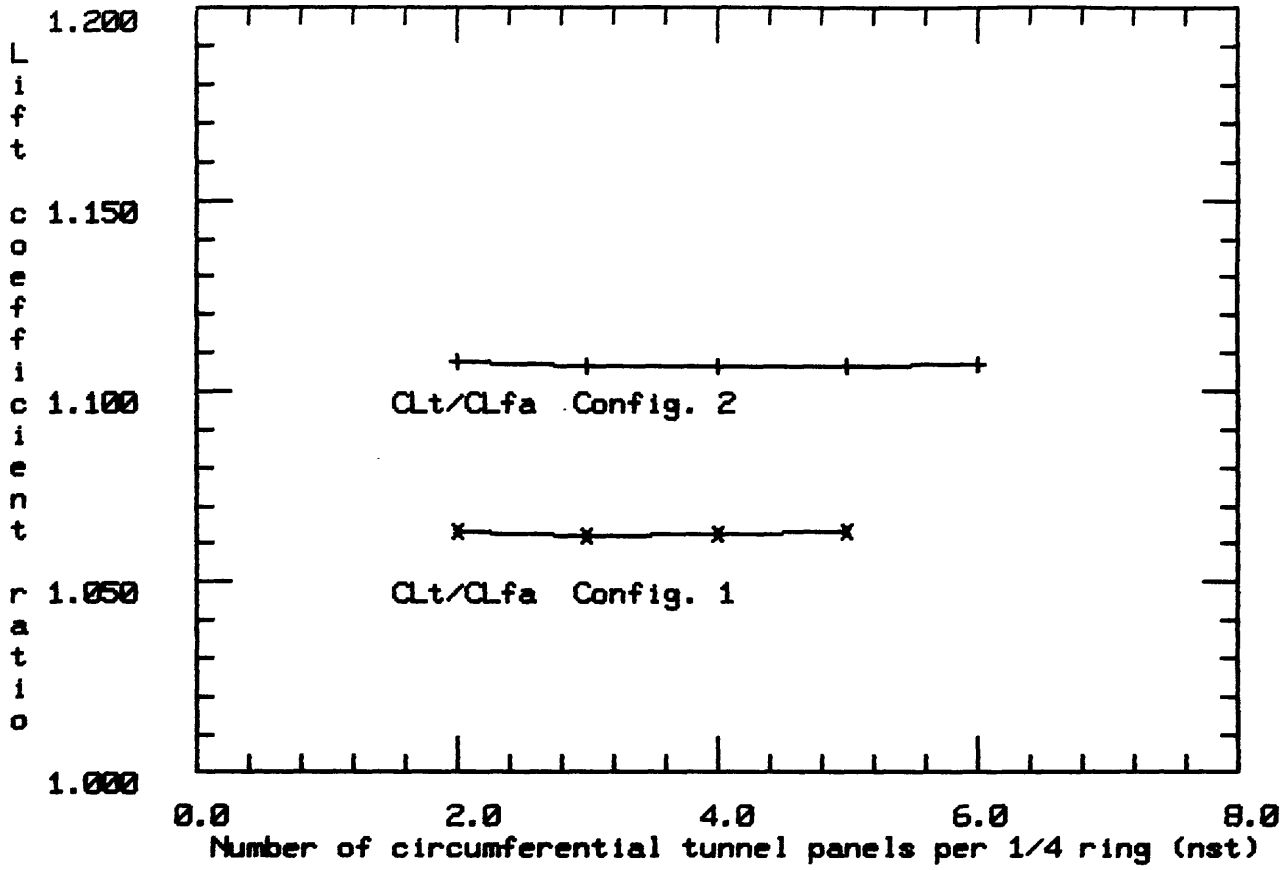


Figure 6-10 Lift coefficient ratio versus number of circumferential tunnel panels

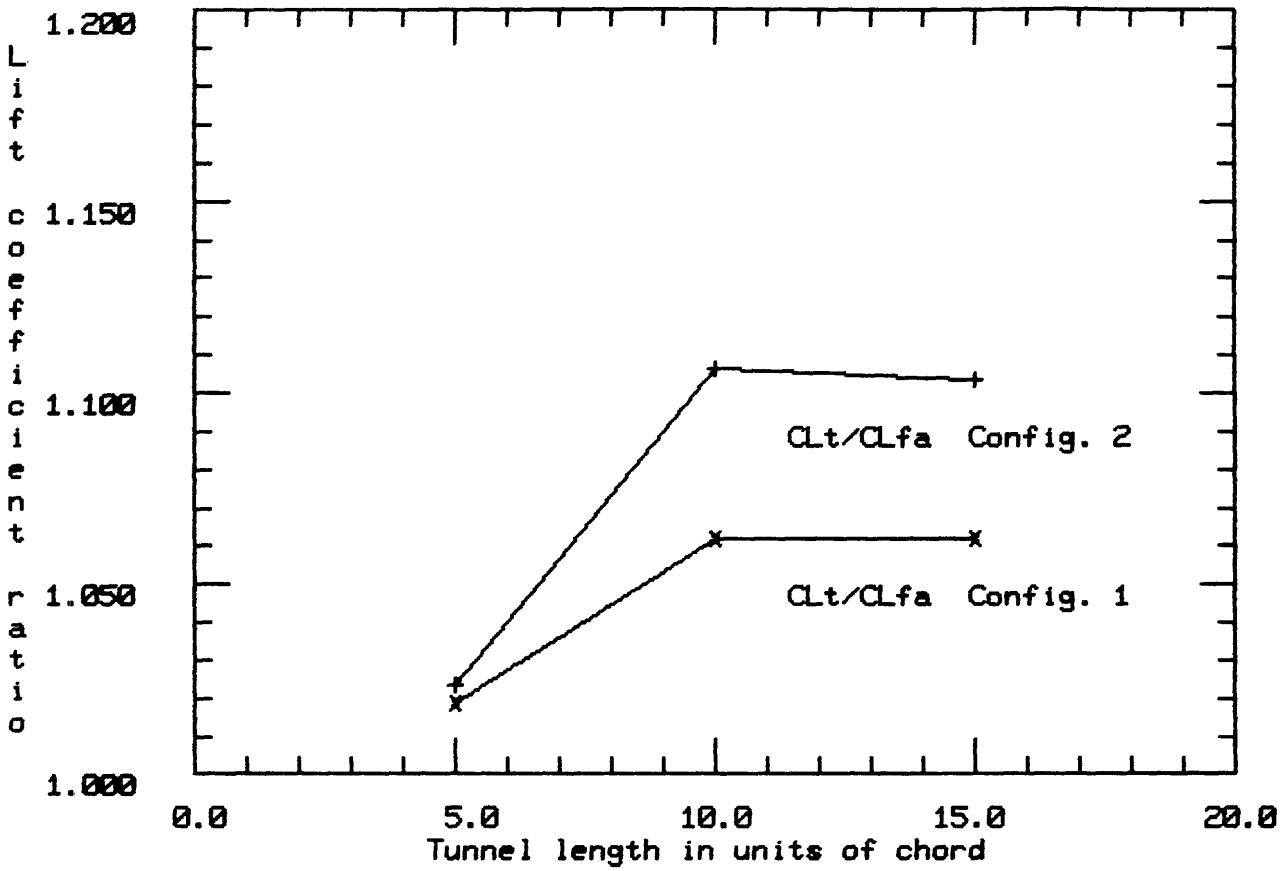


Figure 6-11 Lift coefficient ratio versus tunnel length

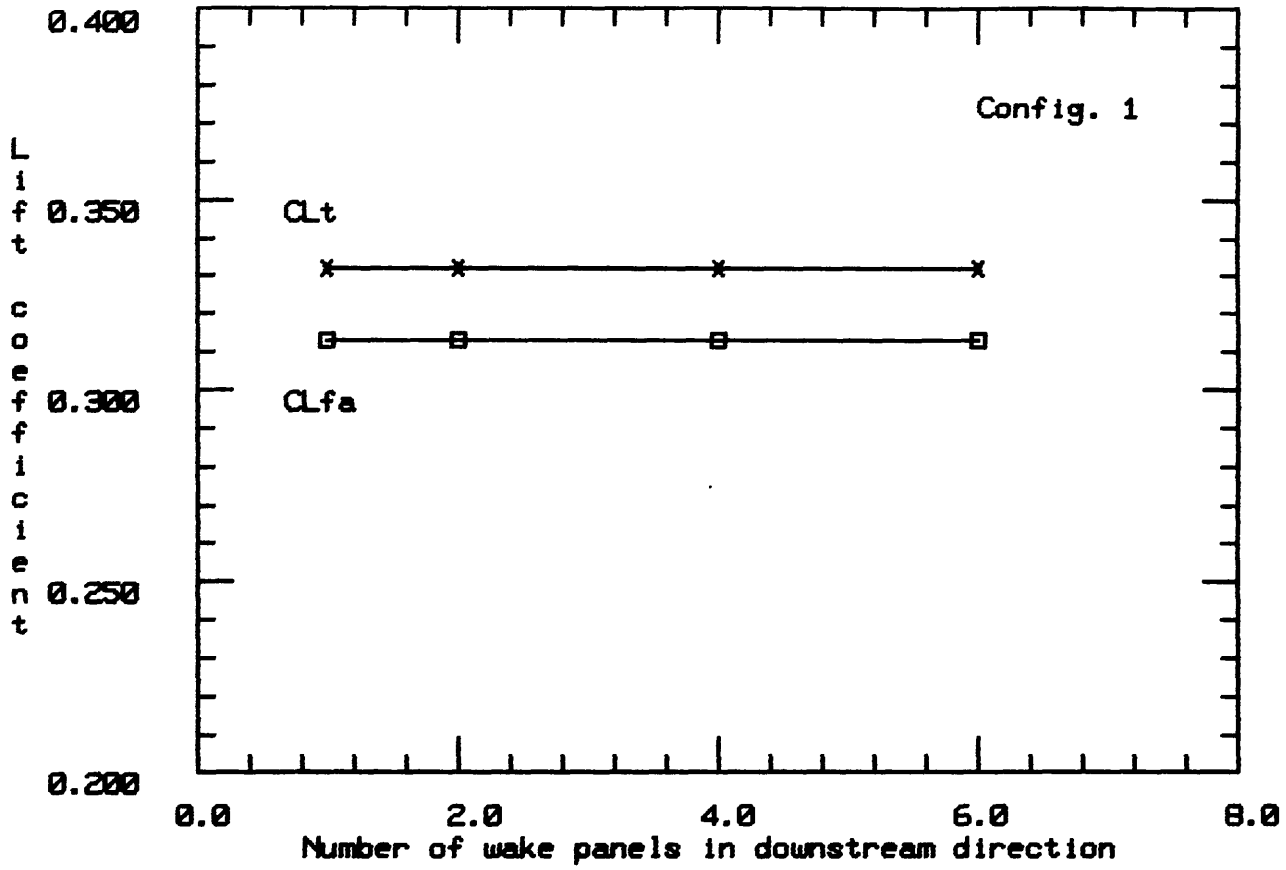


Figure 6-12 Lift coefficient versus number of wing wake panels in downstream direction

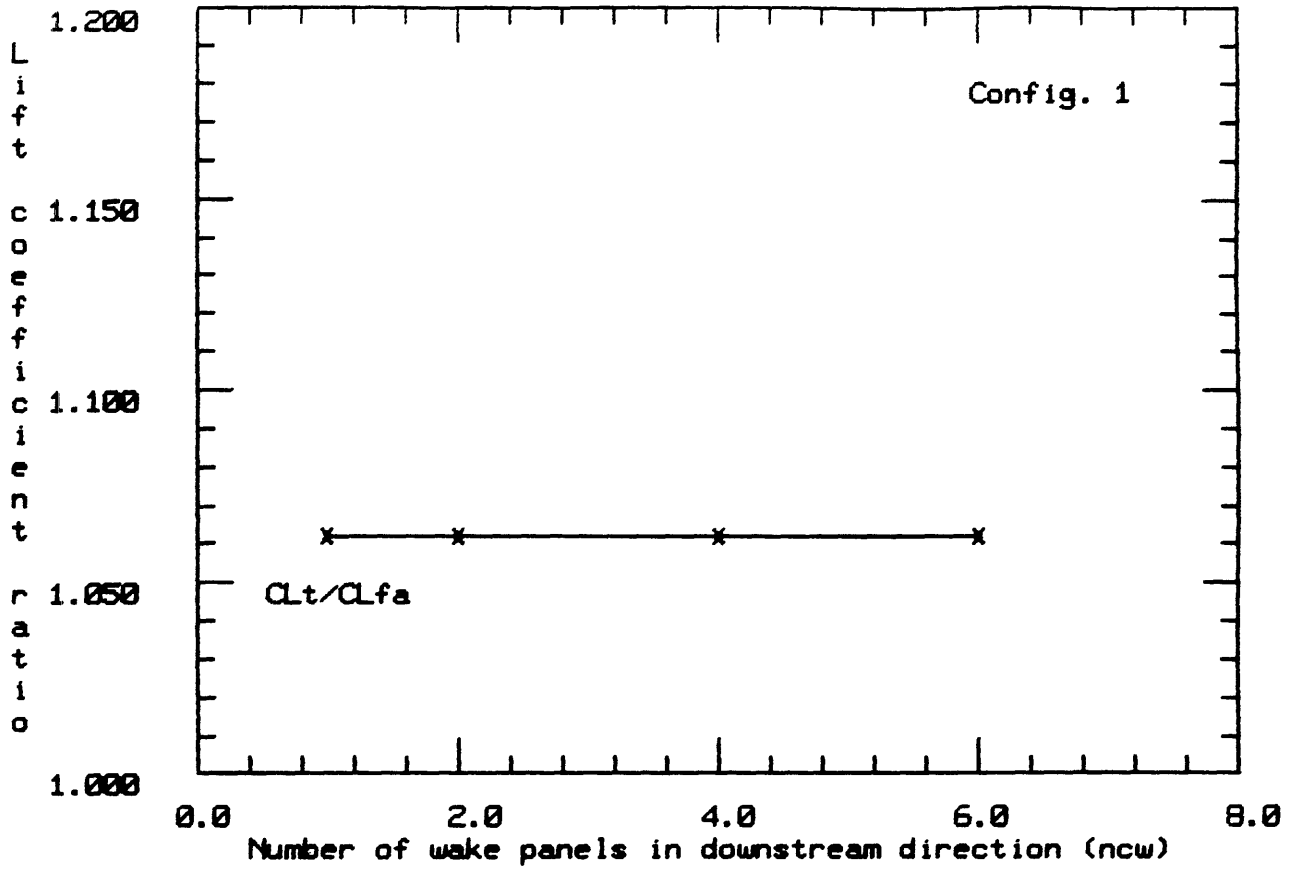


Figure 6-13 Lift coefficient ratio versus number of wake panels in downstream direction

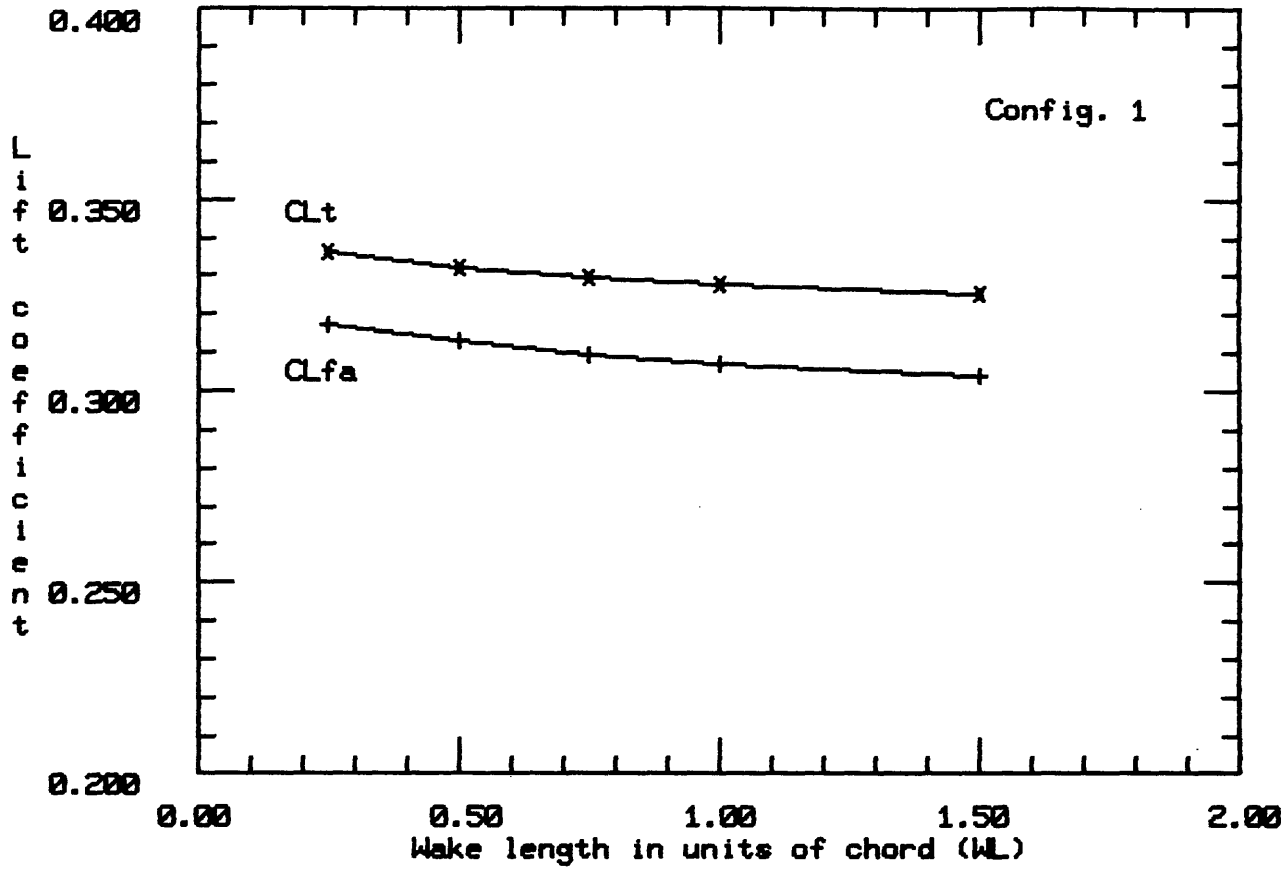


Figure 6-14 Lift coefficient versus wake length for attached flow

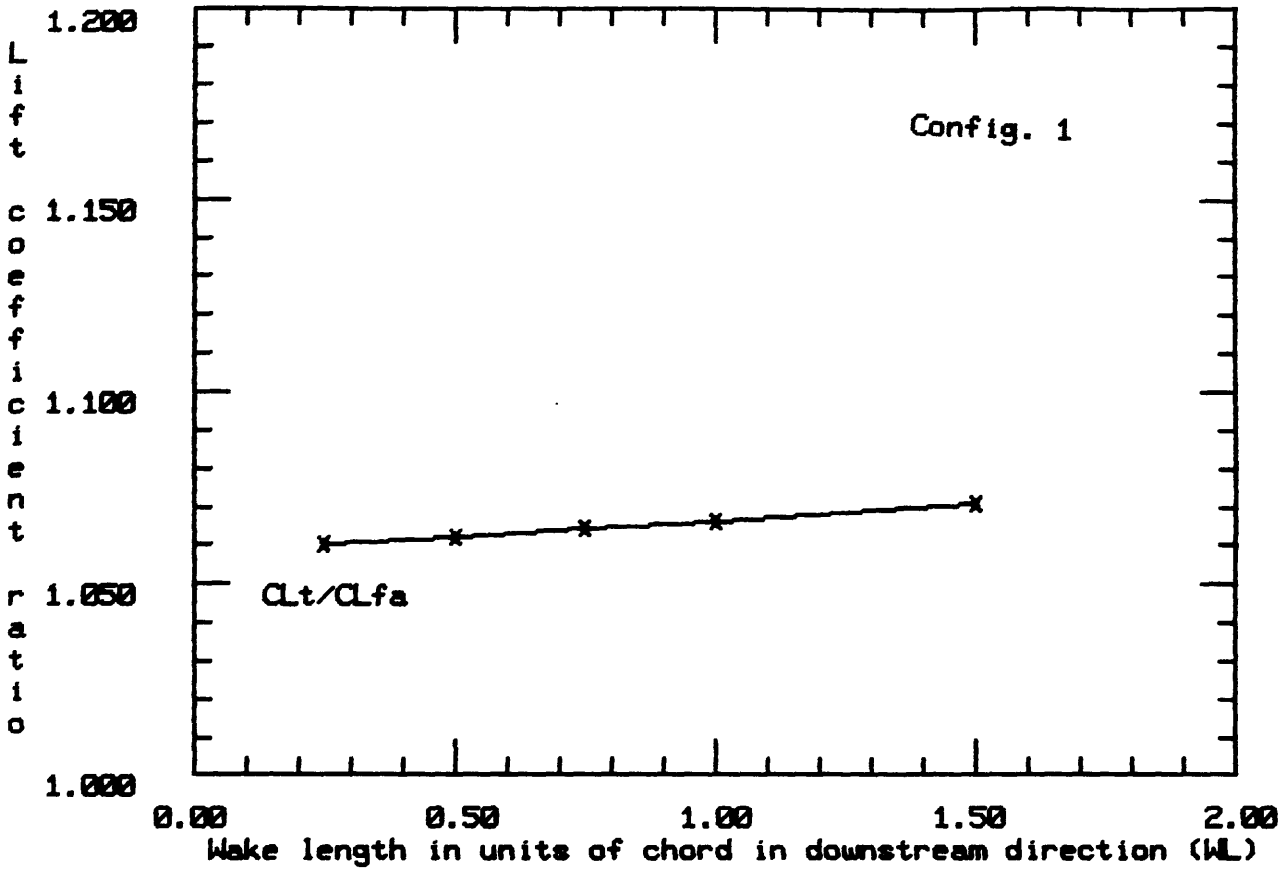


Figure 6-15 Lift coefficient ratio versus wake length for attached flow

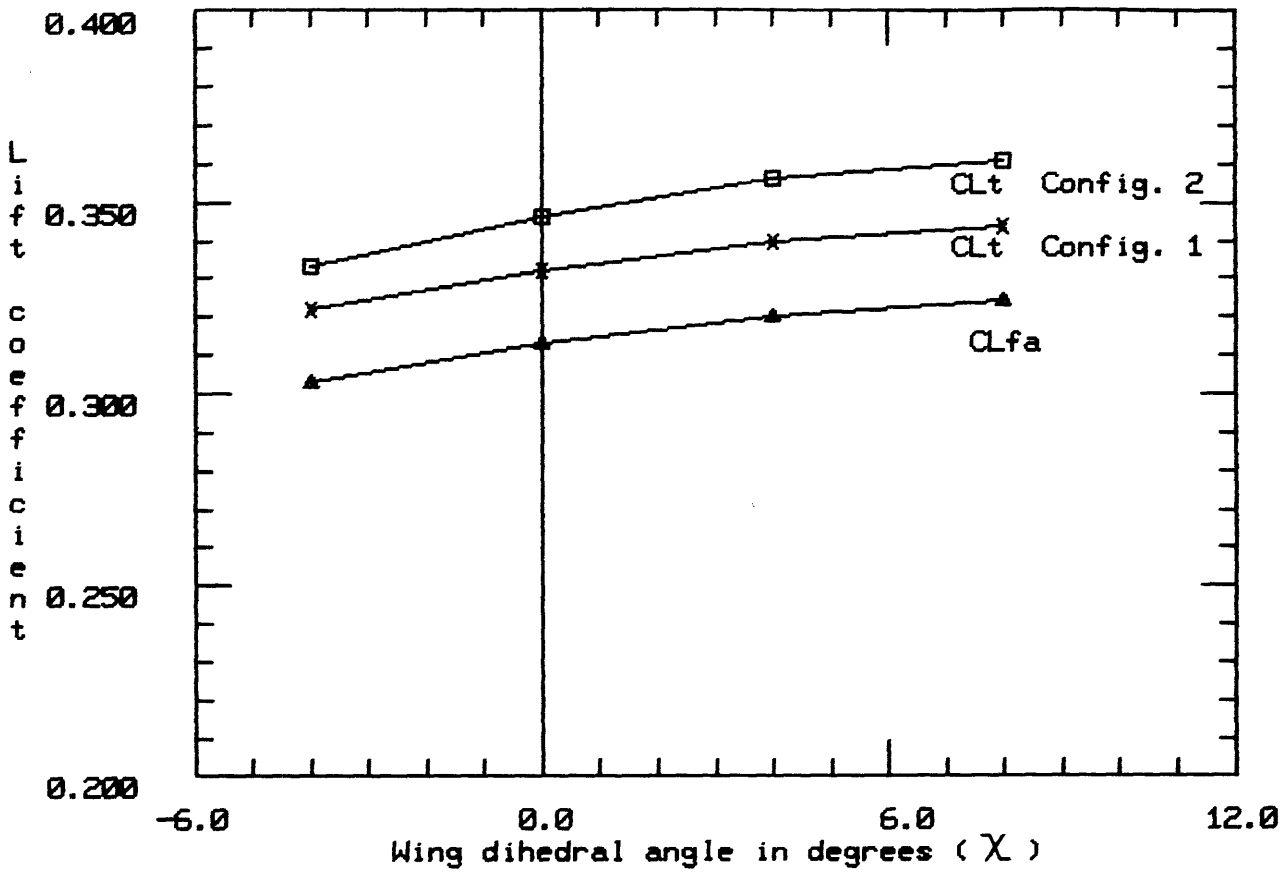


Figure 6-16 Lift coefficient versus wing dihedral angle

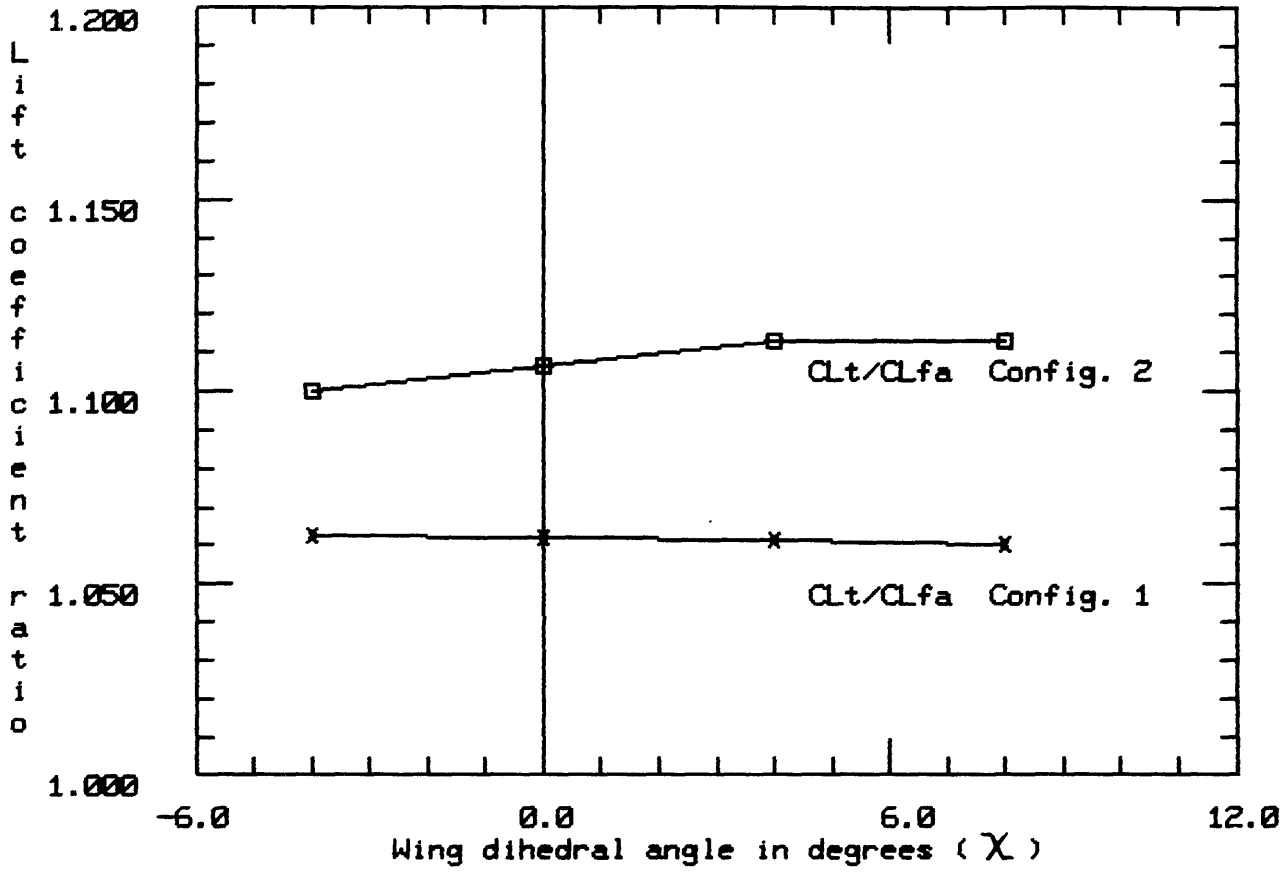


Figure 6-17 Lift coefficient ratio versus wing dihedral angle

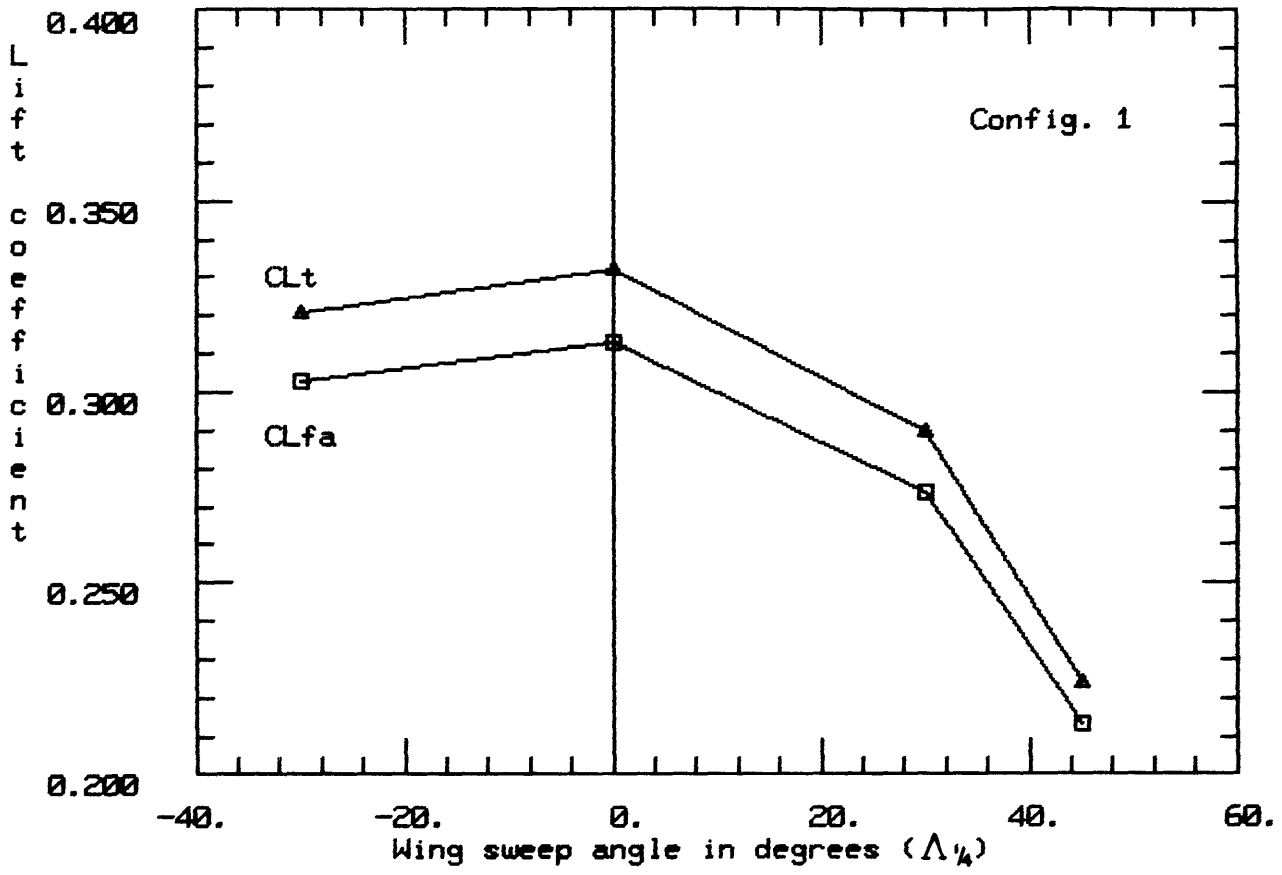


Figure 6-18 Lift coefficient versus sweep angle

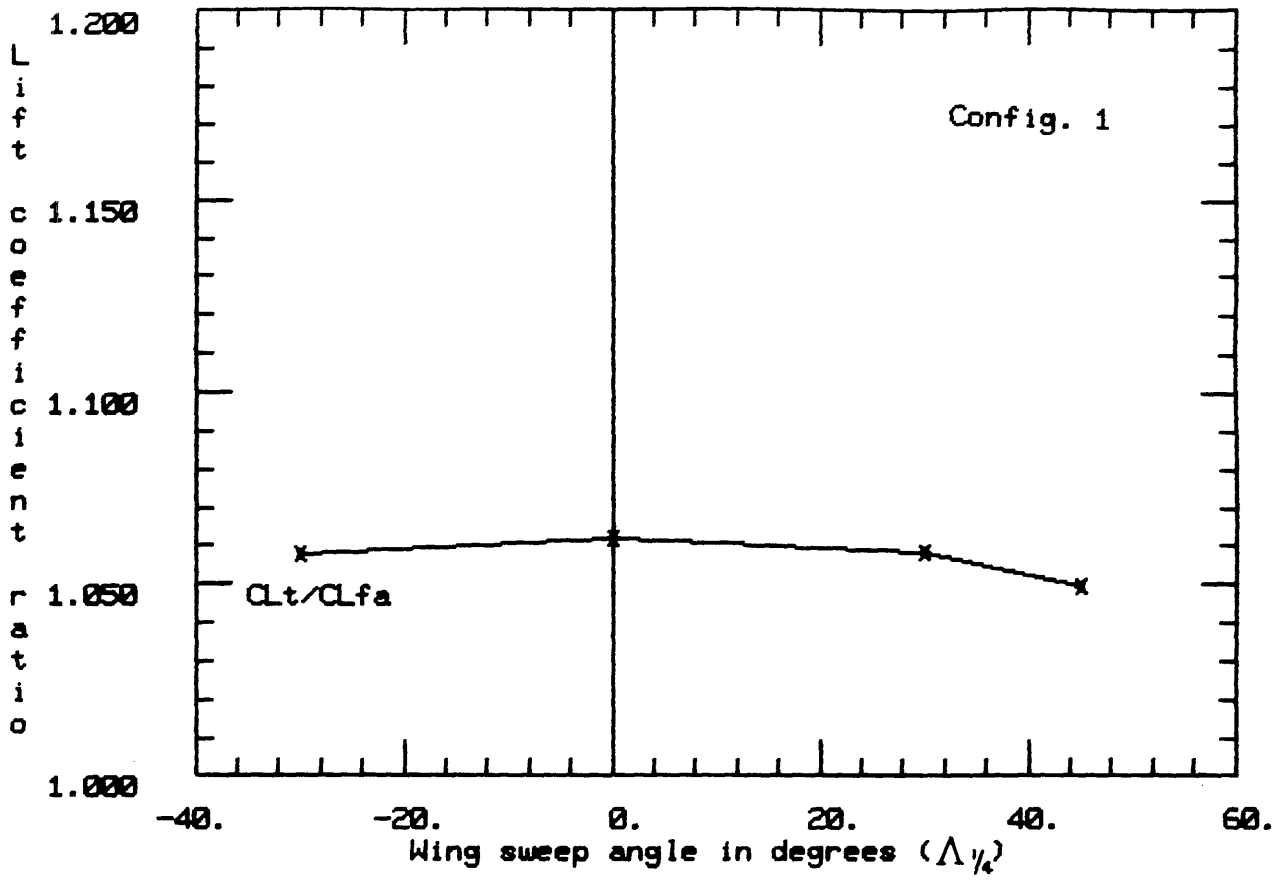


Figure 6-19 Lift coefficient ratio versus sweep angle

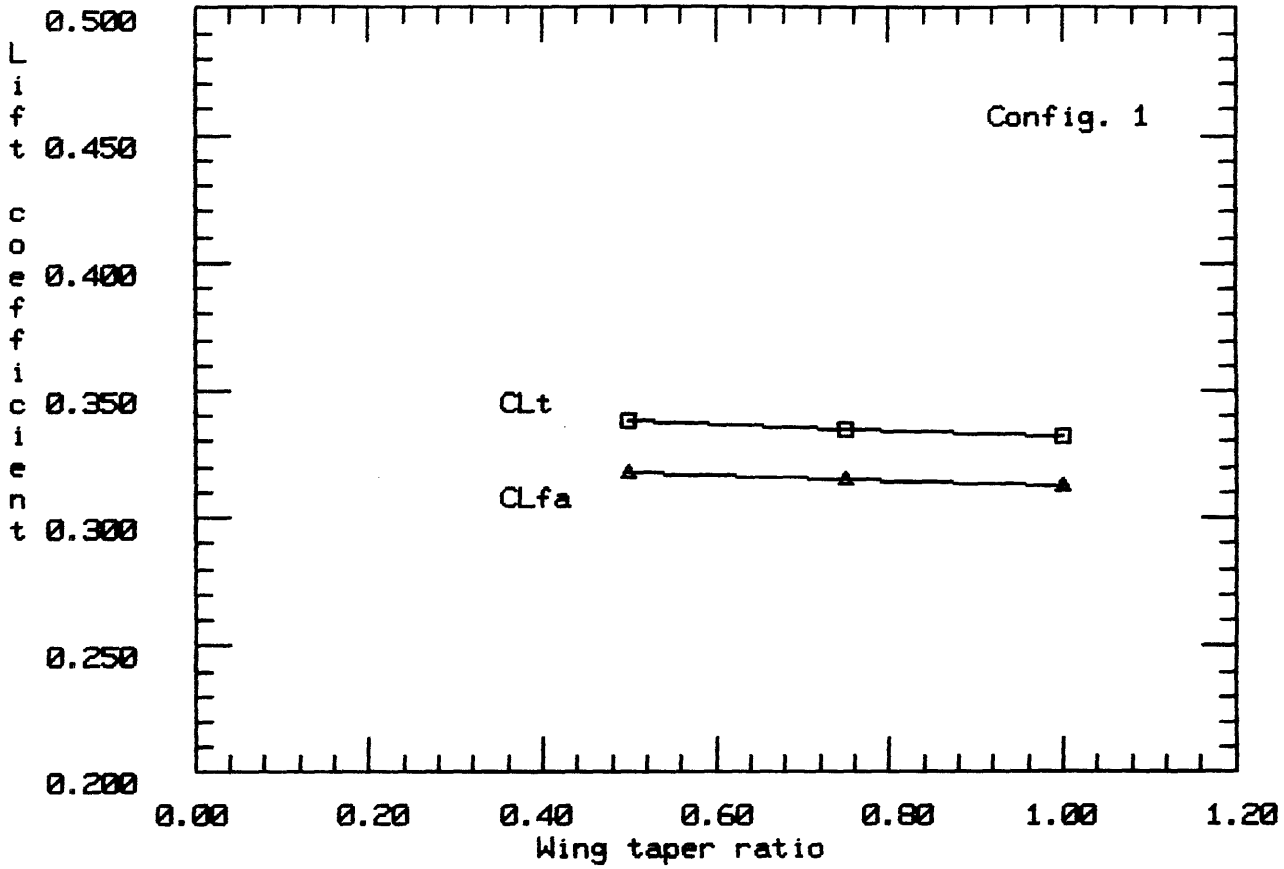


Figure 6-20 Lift coefficient versus wing taper ratio

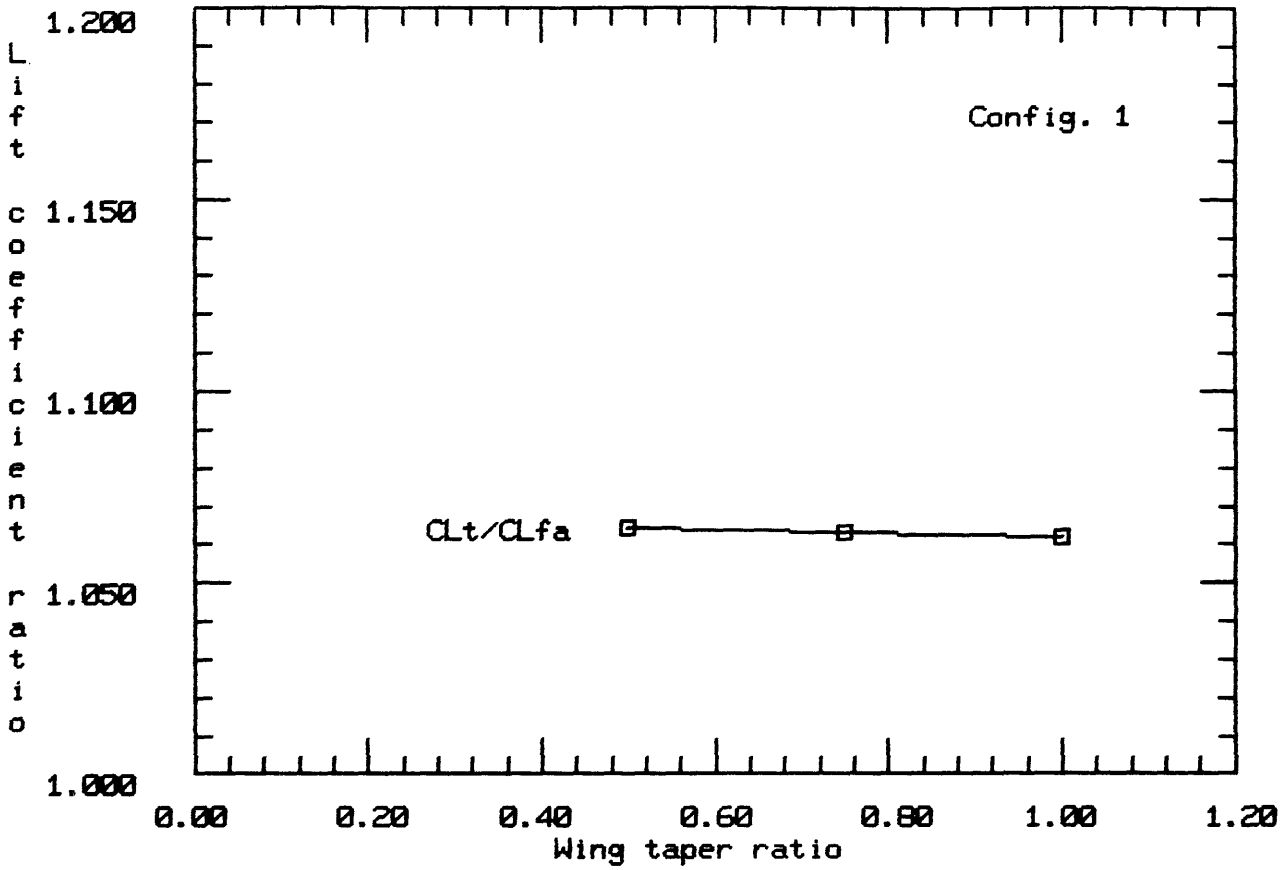


Figure 6-21 Lift coefficient ratio versus wing taper ratio

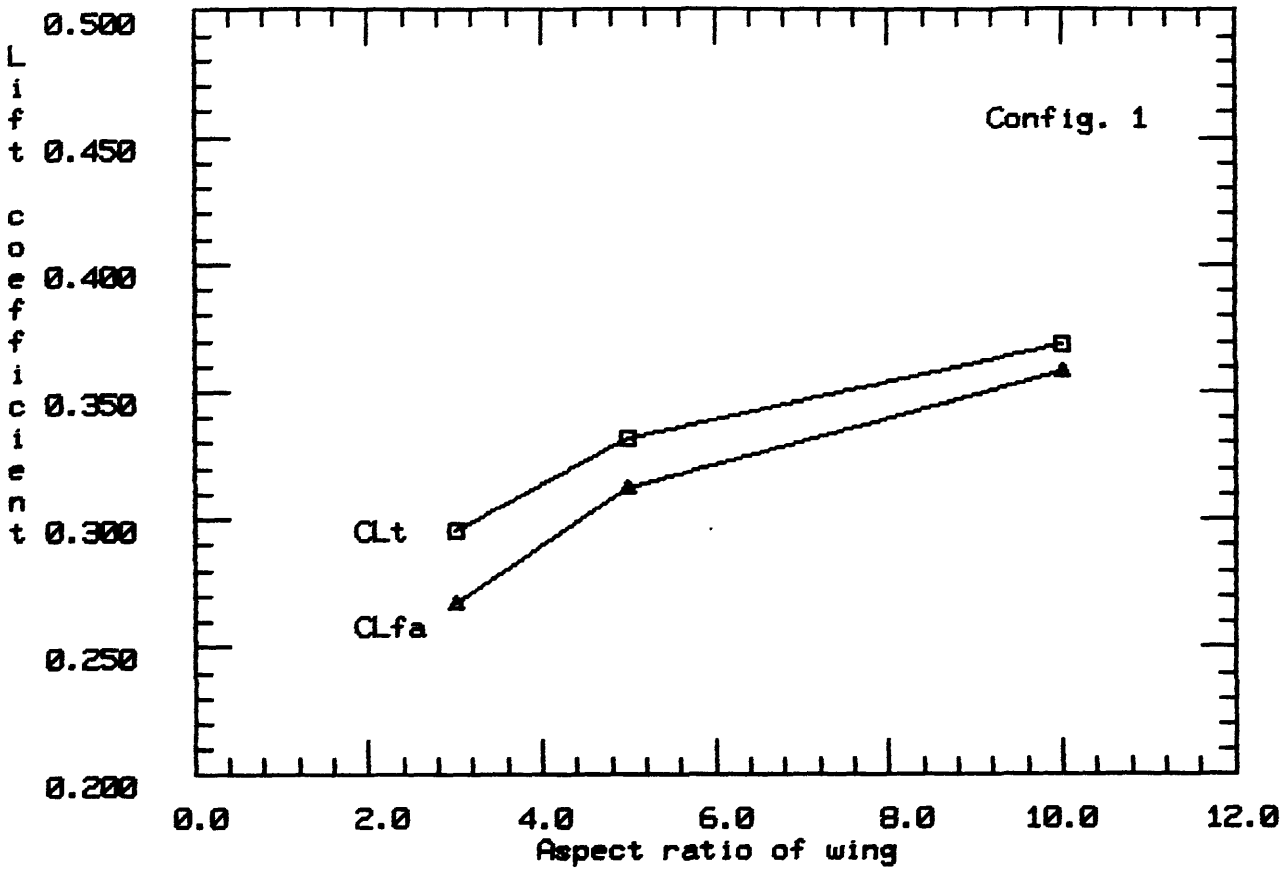


Figure 6-22 Lift coefficient versus wing aspect ratio

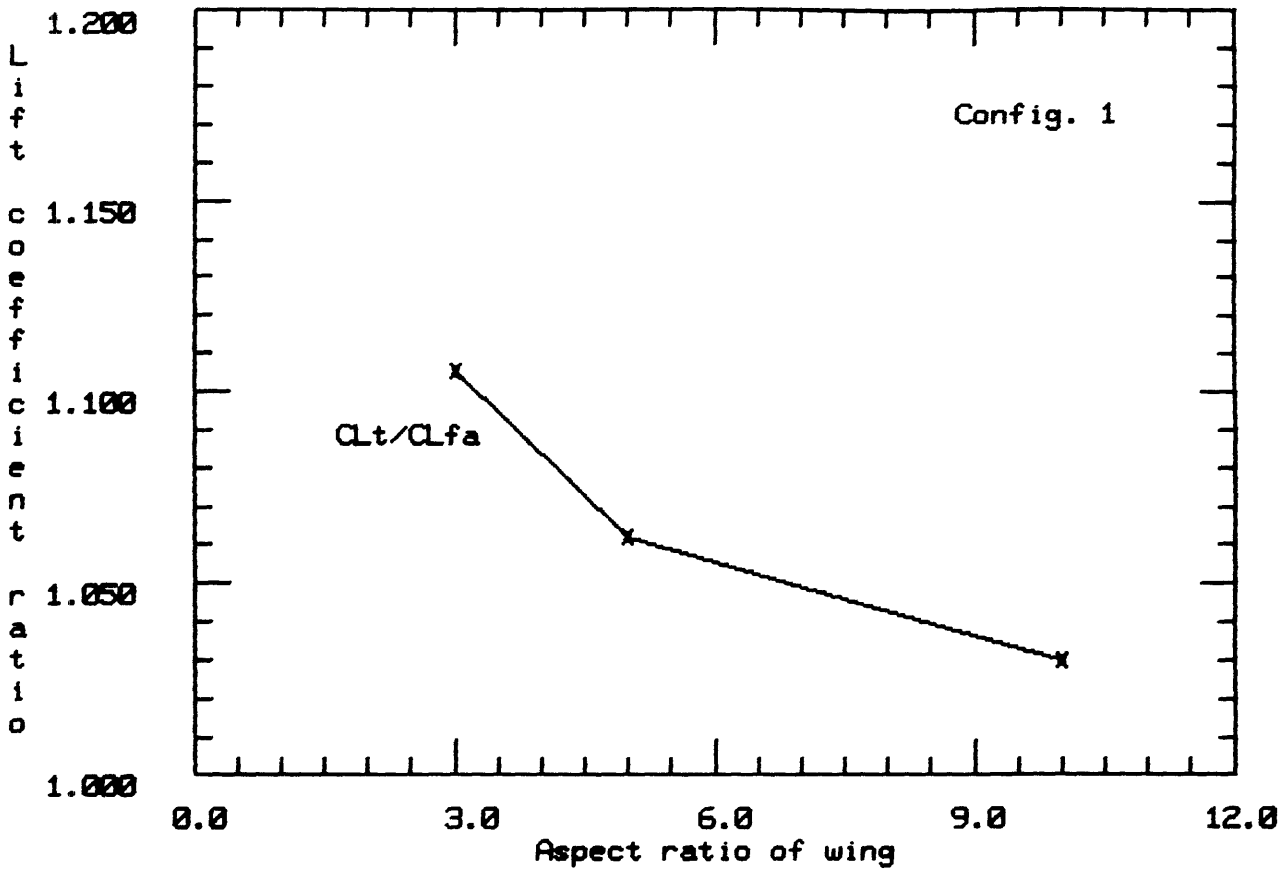


Figure 6-23 Lift coefficient ratio versus wing aspect ratio

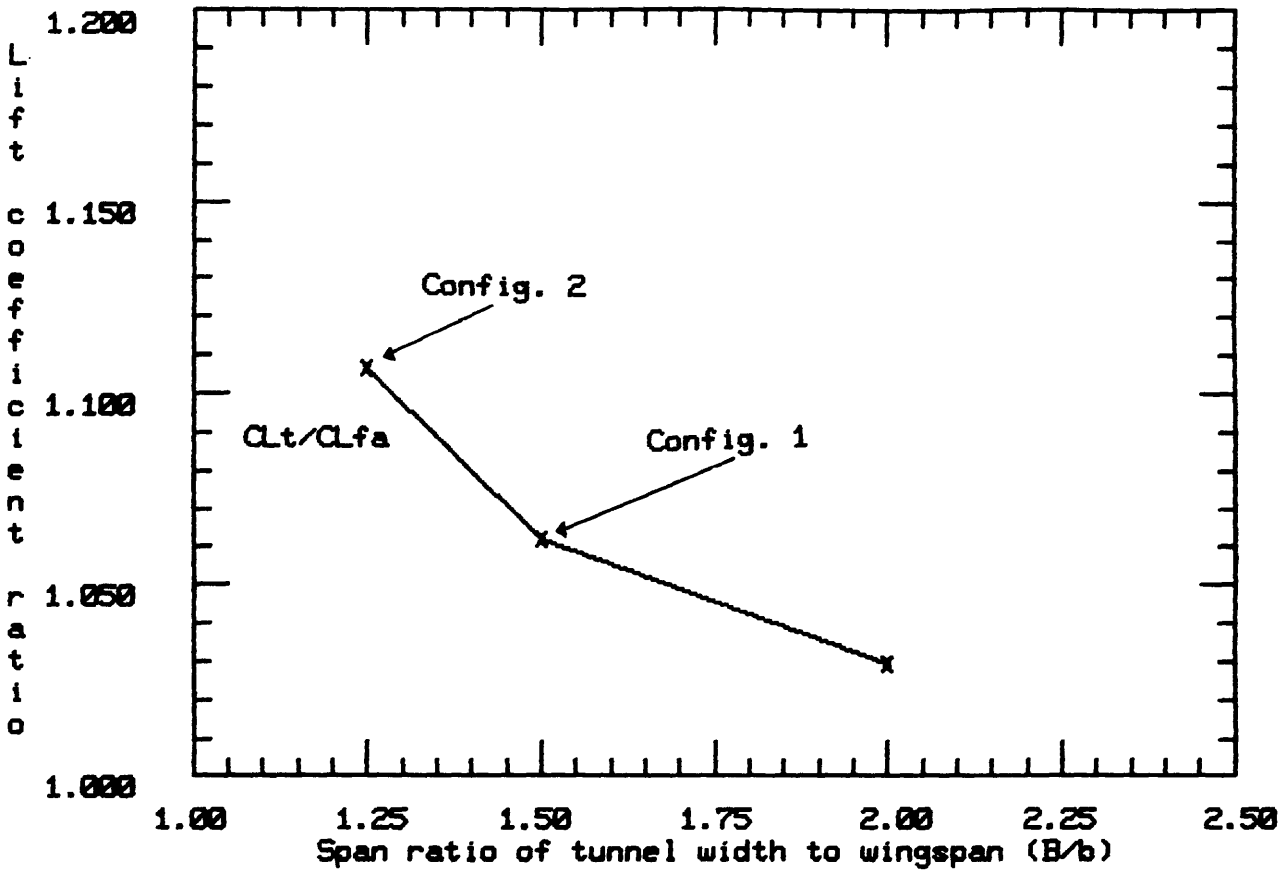


Figure 6-24 Lift coefficient ratio versus span ratio (B/b)

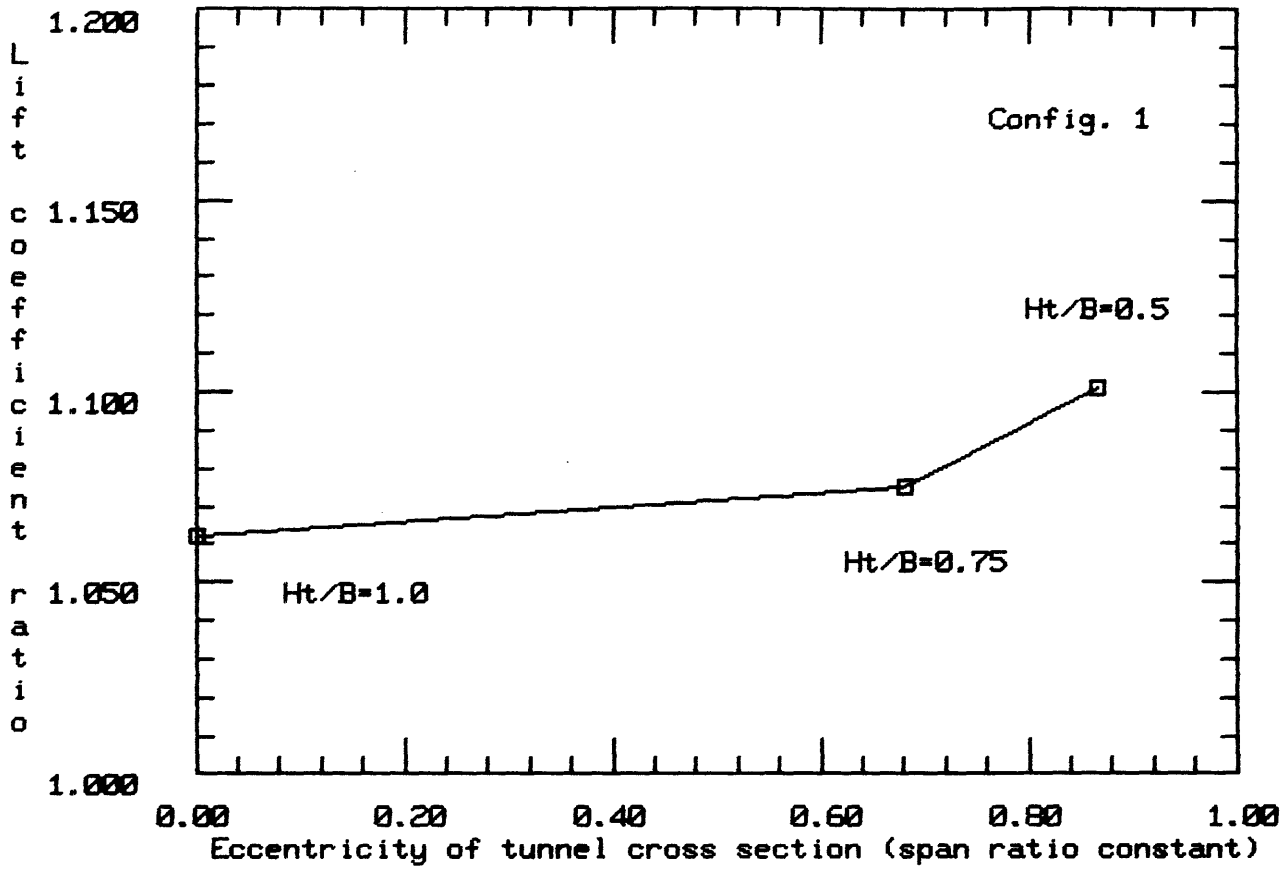


Figure 6-25 Lift coefficient ratio versus tunnel eccentricity (span ratio constant)

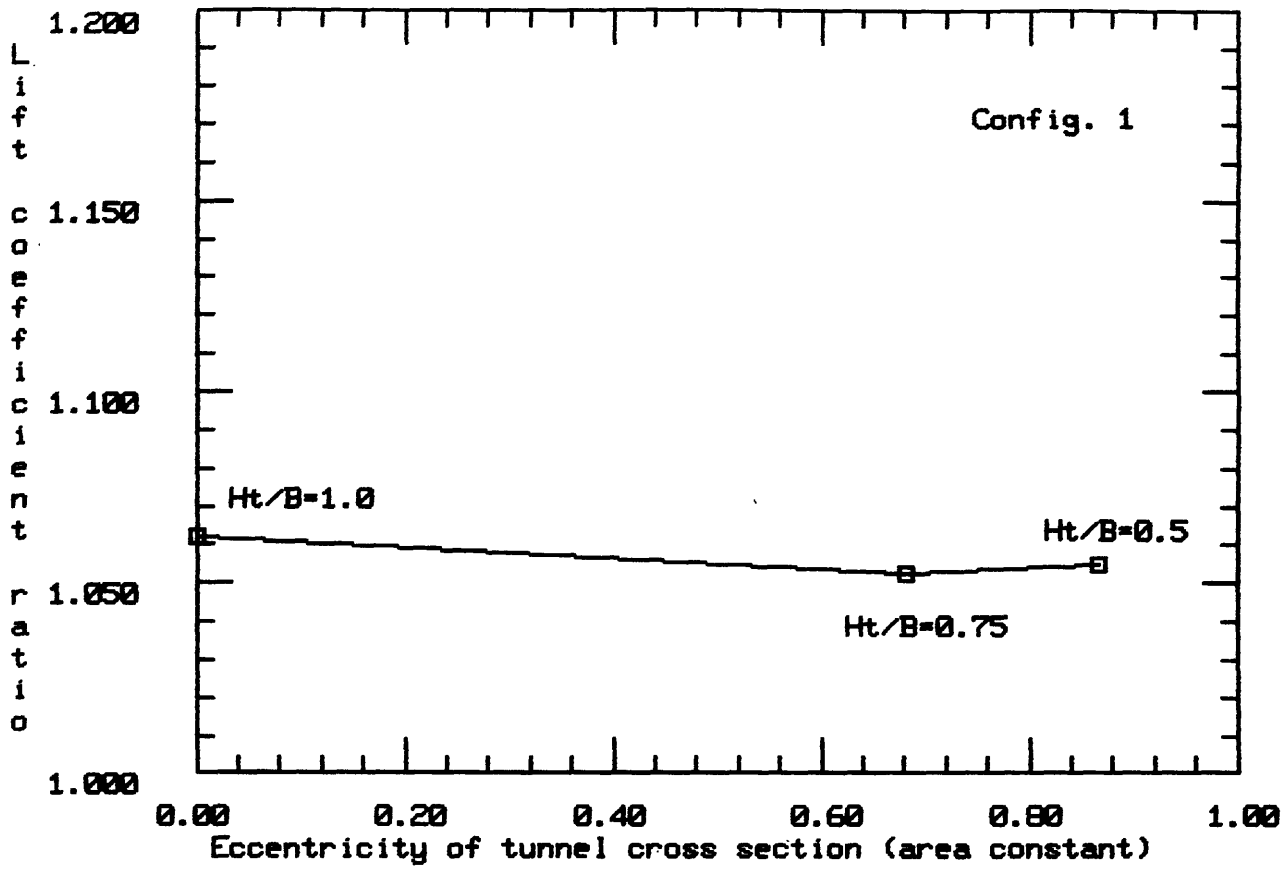


Figure 6-26 Lift coefficient ratio versus tunnel eccentricity (tunnel area constant)

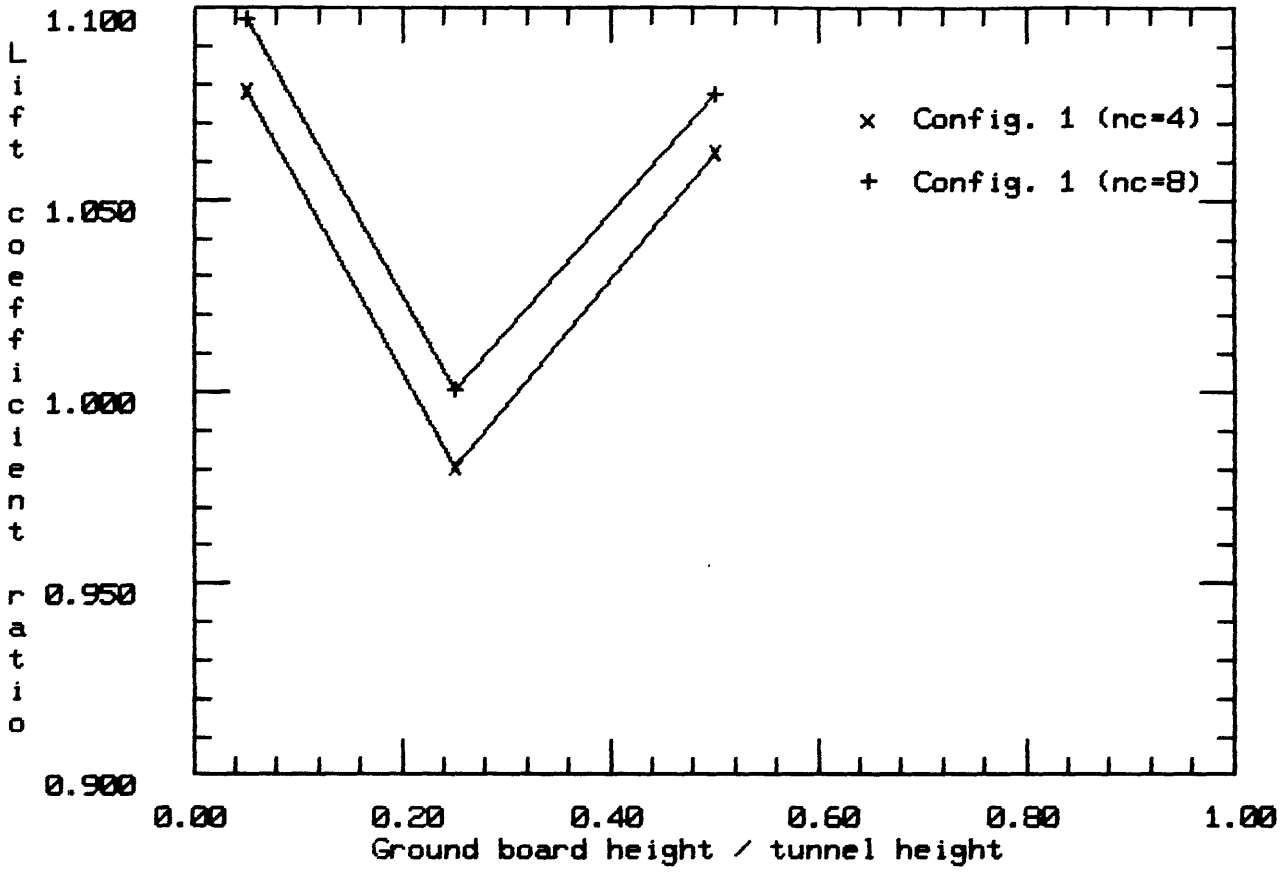


Figure 6-27 Lift coefficient ratio versus ground board height

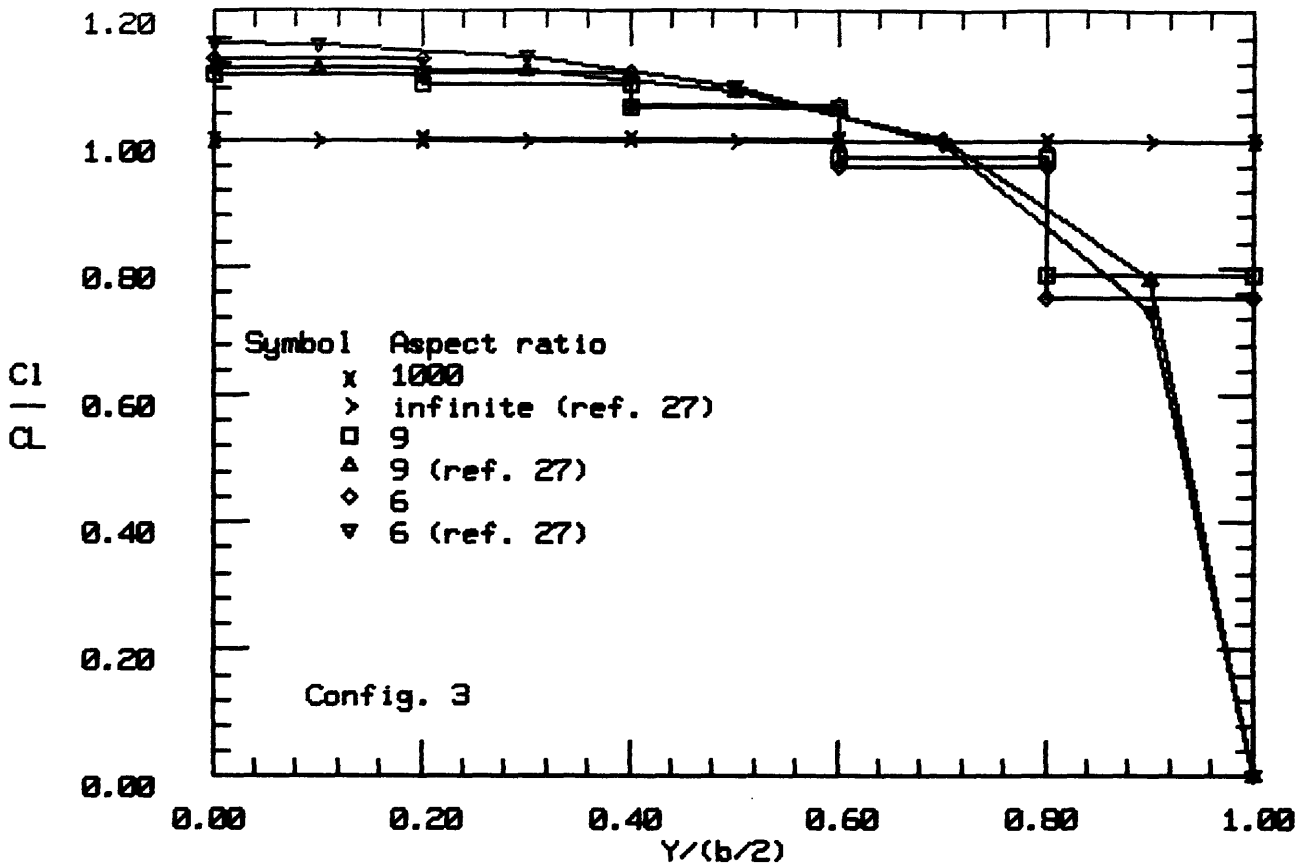


Figure 6-28 Spanwise lift distribution versus wing aspect ratio

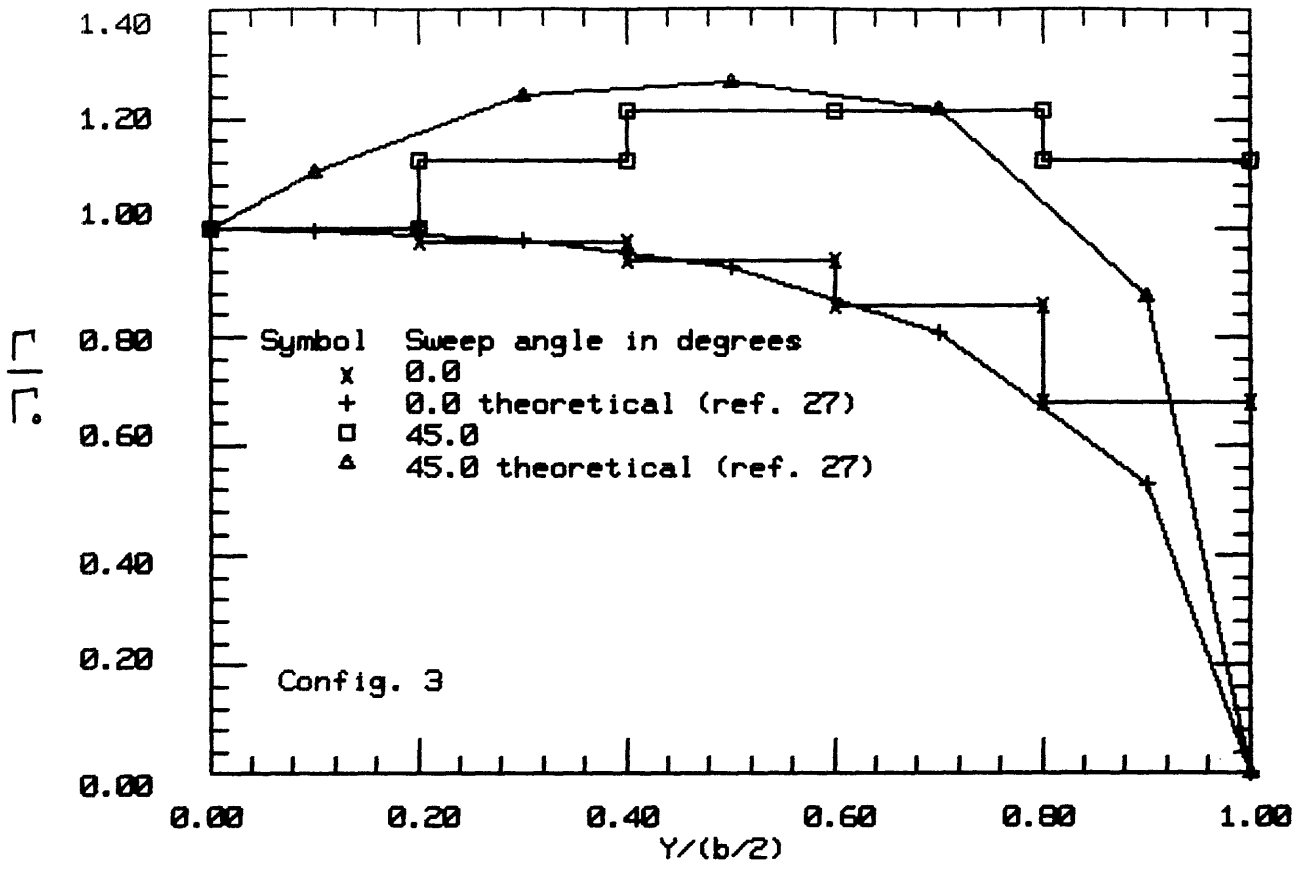


Figure 6-29 Spanwise circulation distribution versus wing sweep angle

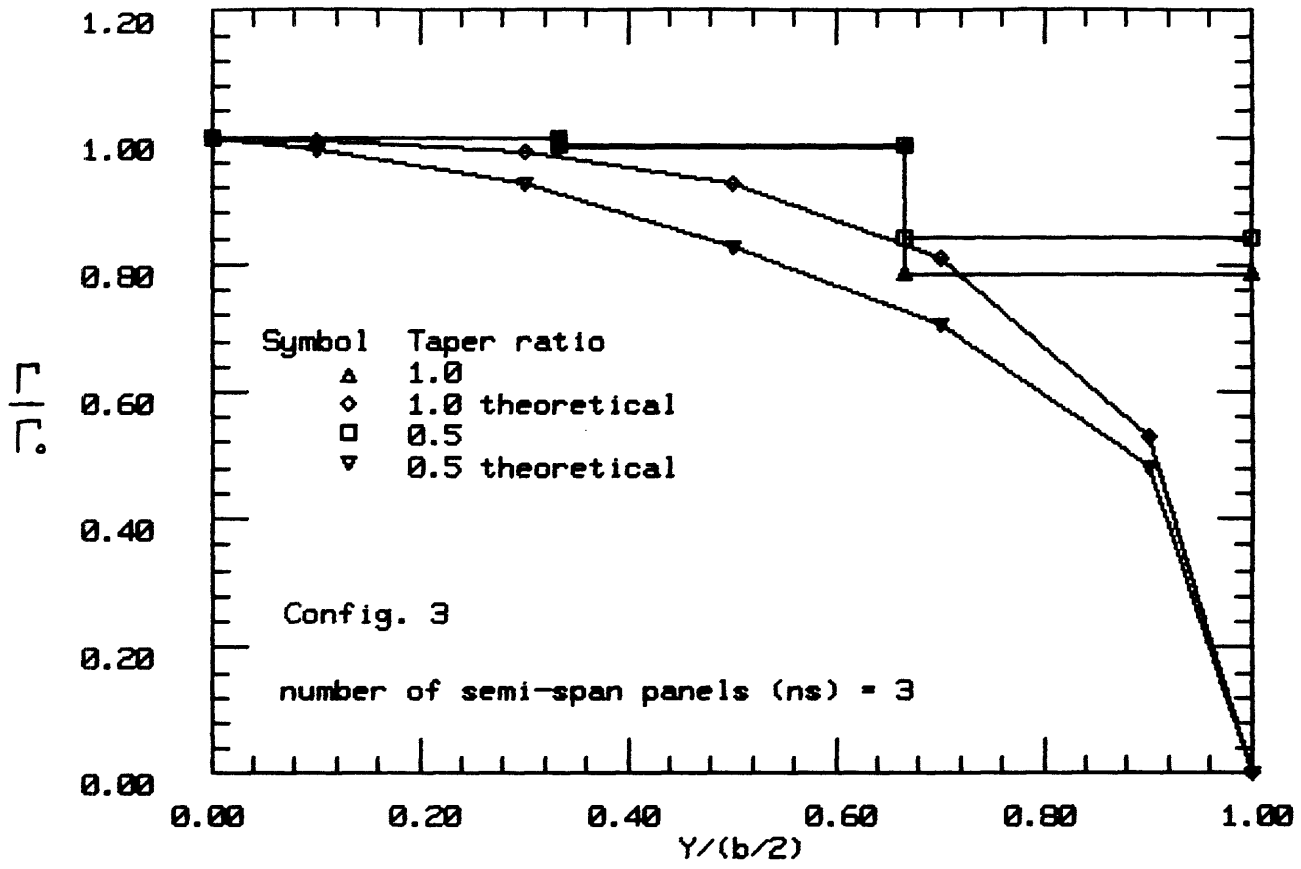


Figure 6-30 Spanwise circulation distribution versus wing taper ratio

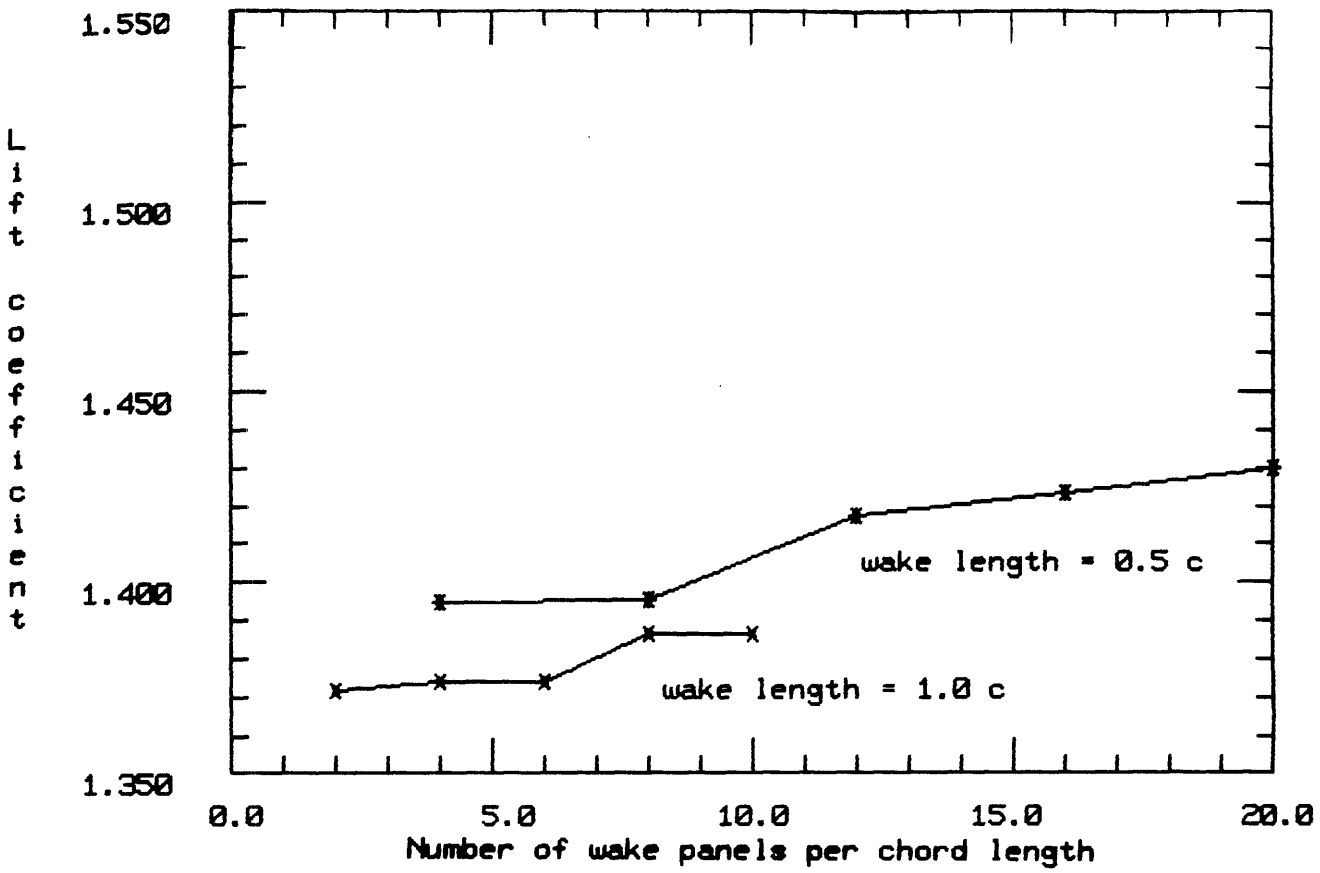


Figure 6-31 Coefficient of lift versus wake panel density

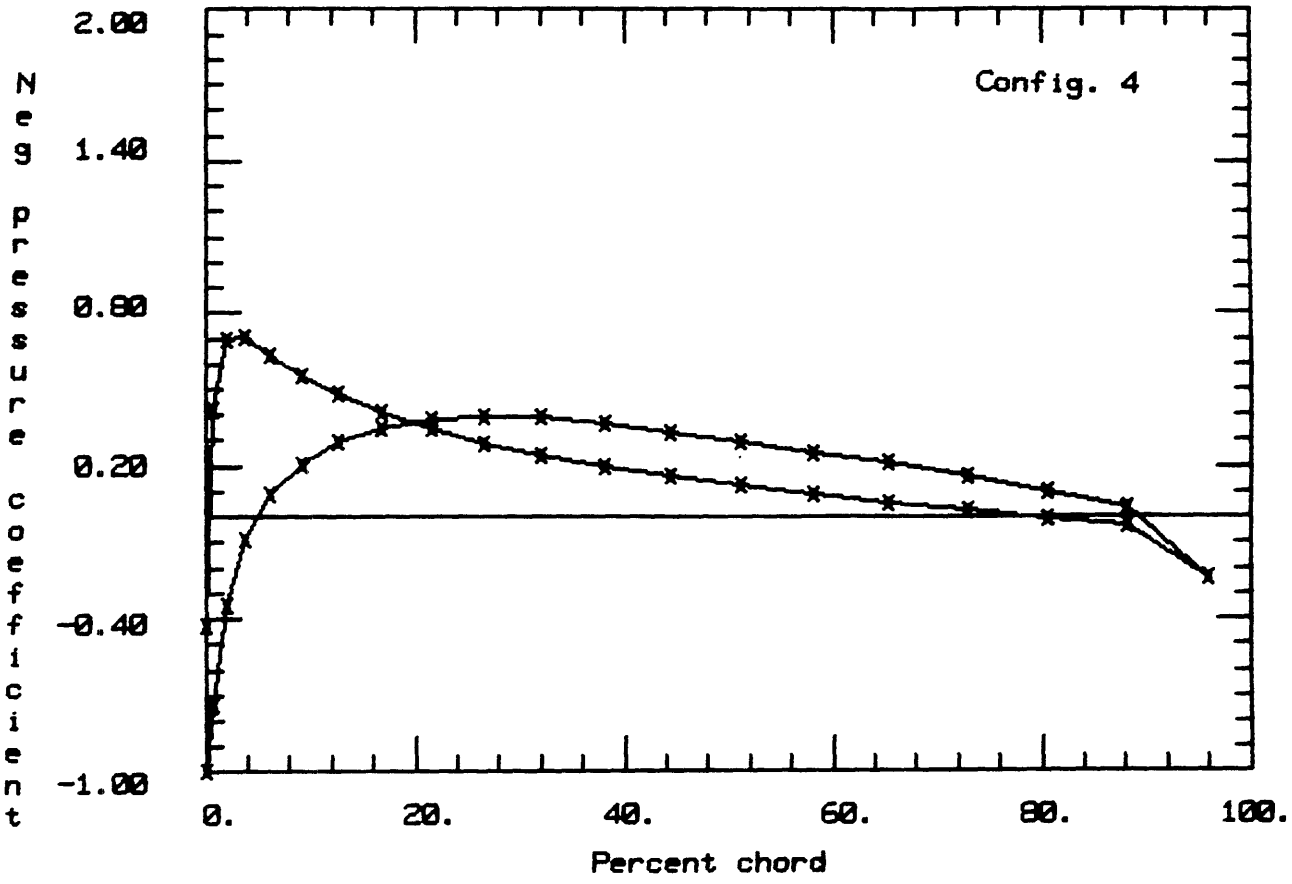


Figure 6-32 Pressure distribution of wing 4 at alpha = -1.7 deg

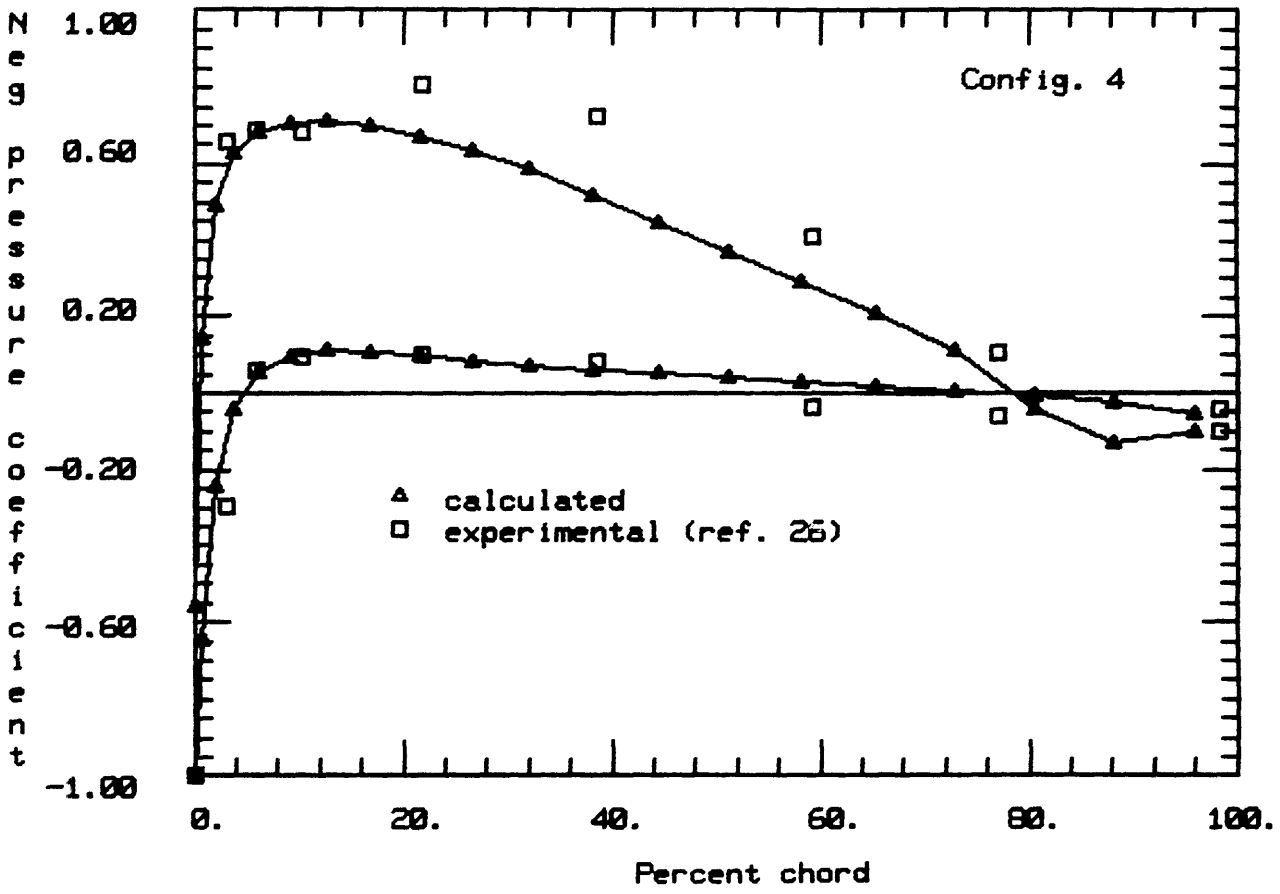


Figure 6-33 Pressure distribution of wing 4 at alpha = 2.8 deg

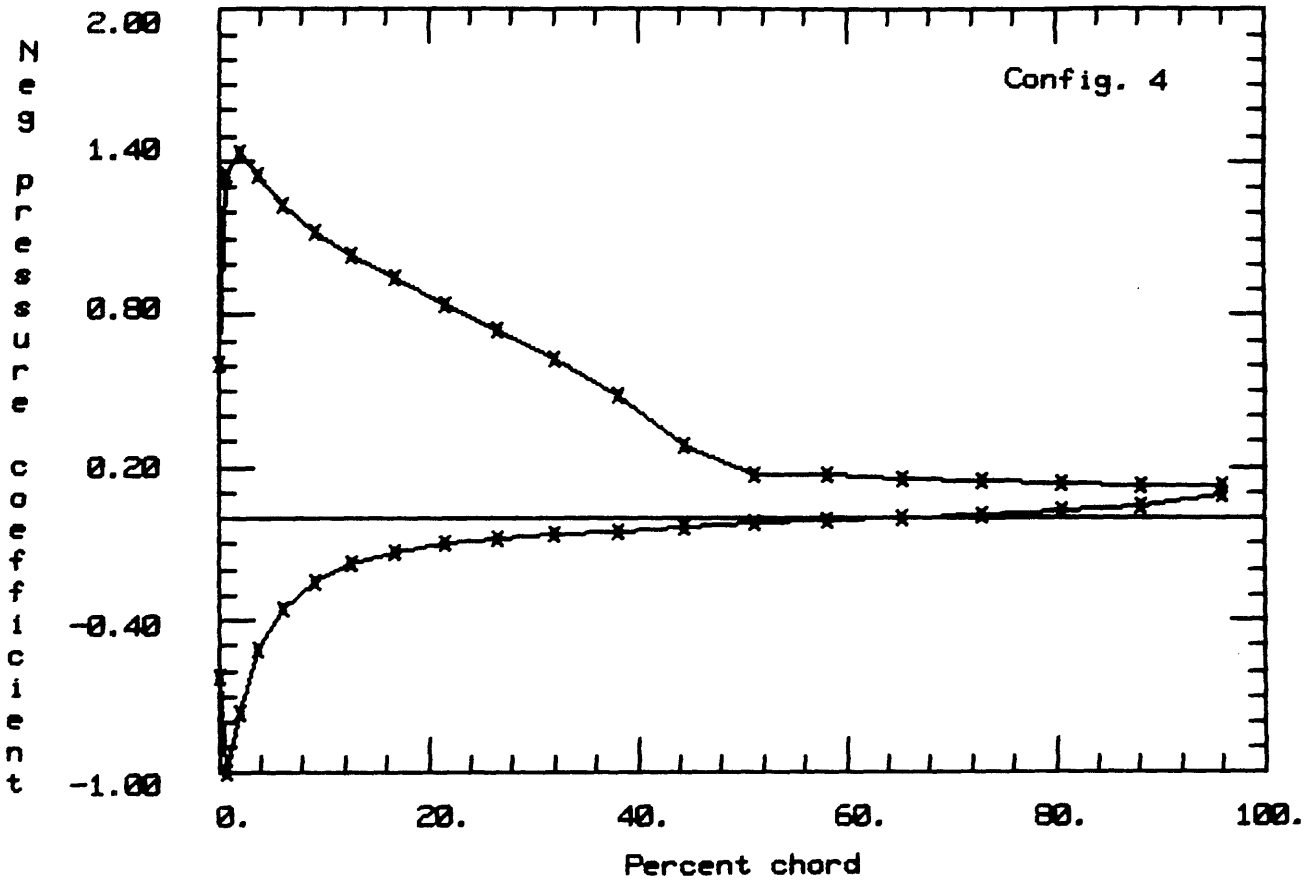


Figure 6-34 Pressure distribution of wing 4 at alpha = 7.4 deg

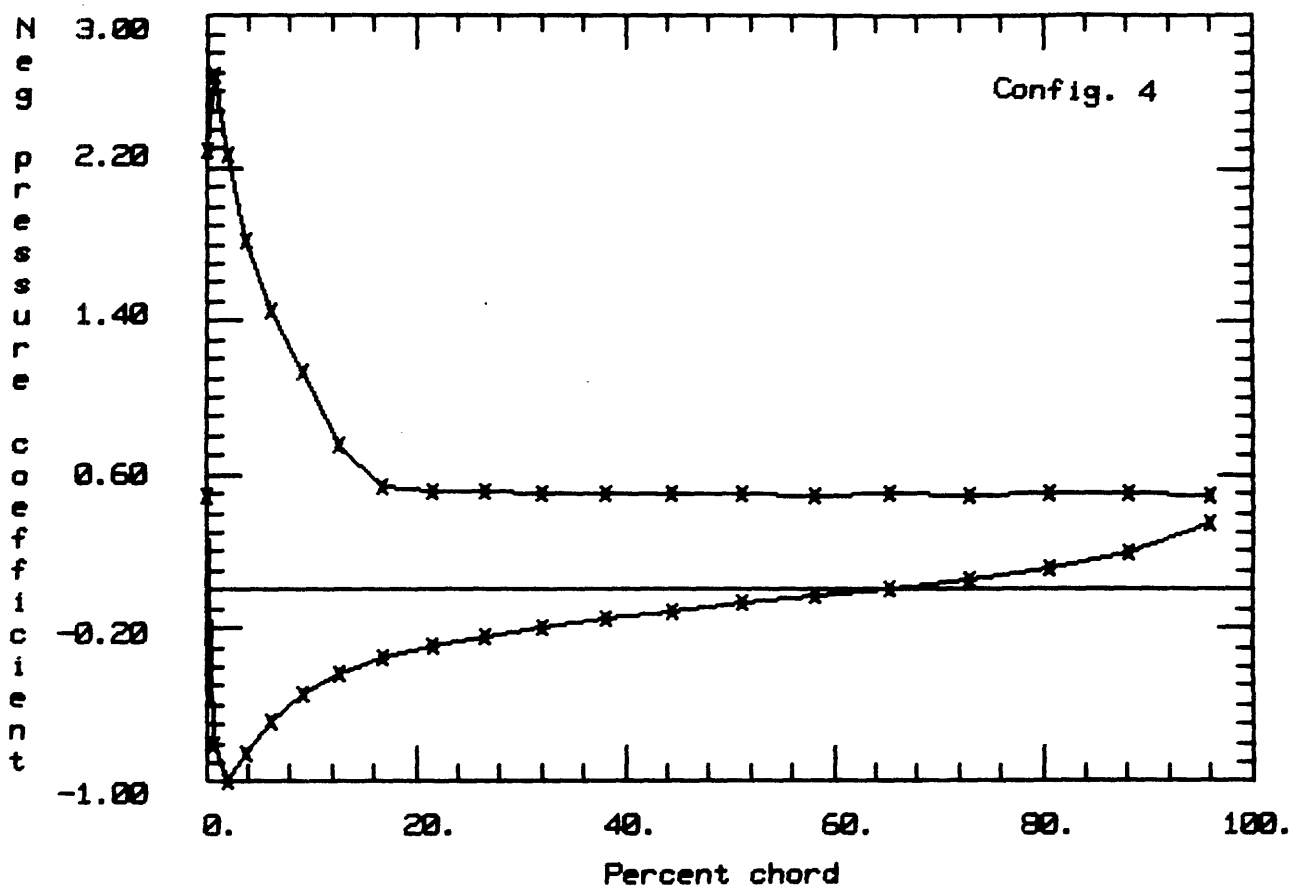


Figure 6-35 Pressure distribution of wing 4 at alpha = 13.8 deg

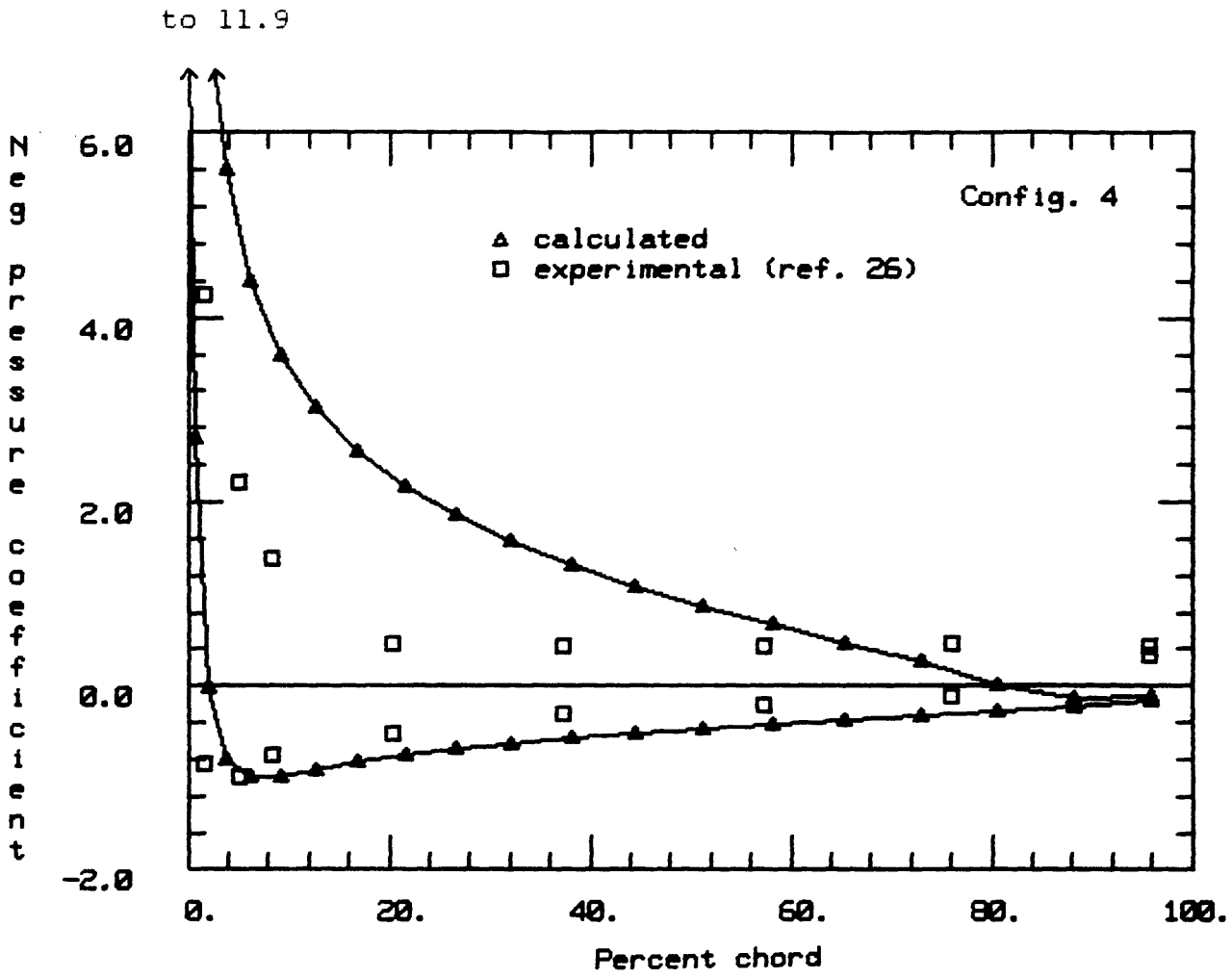


Figure 6-36 Pressure distribution of wing 4 at alpha = 17.9 deg

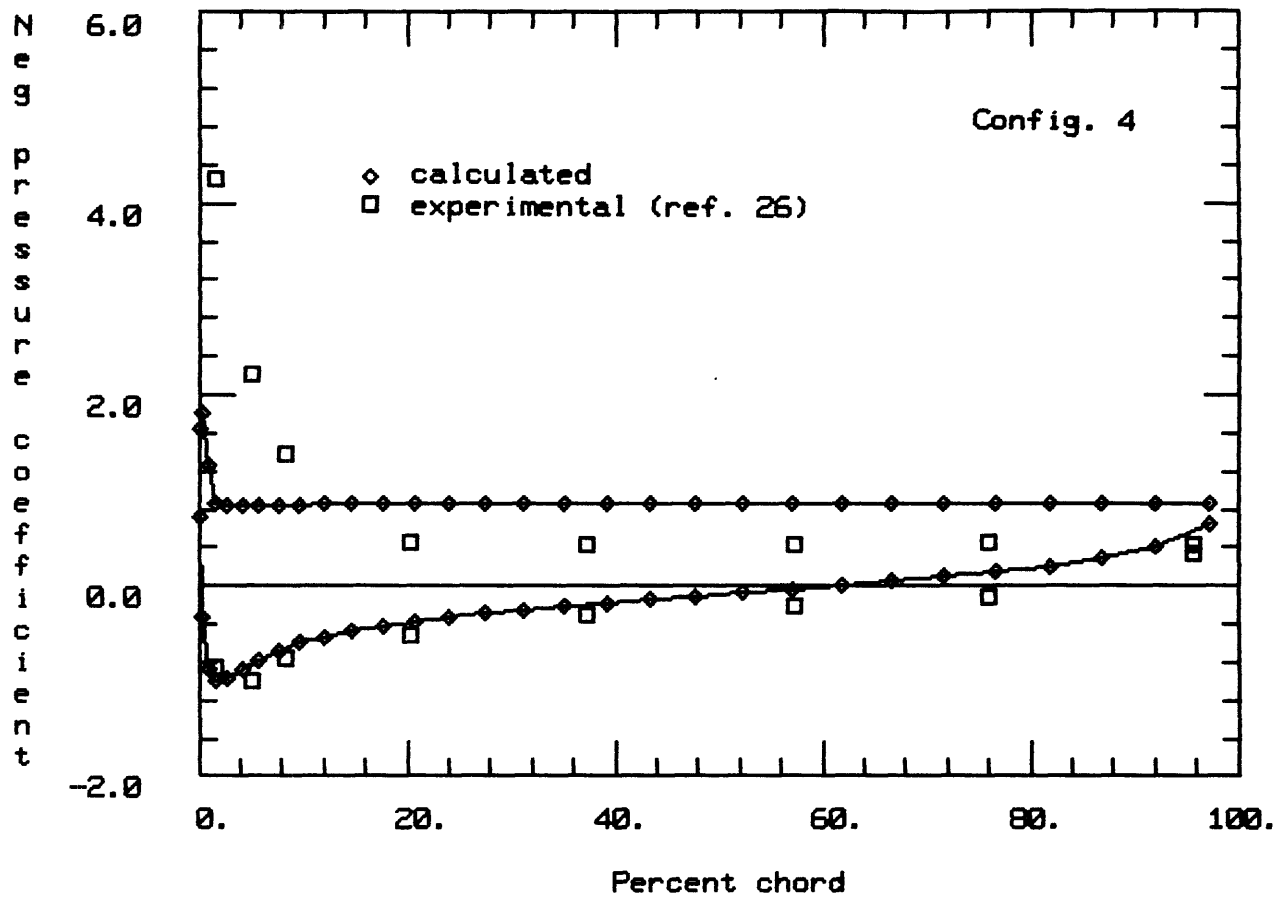


Figure 6-37 Pressure distribution of wing 4 (nc = 30) at alpha = 17.9 deg

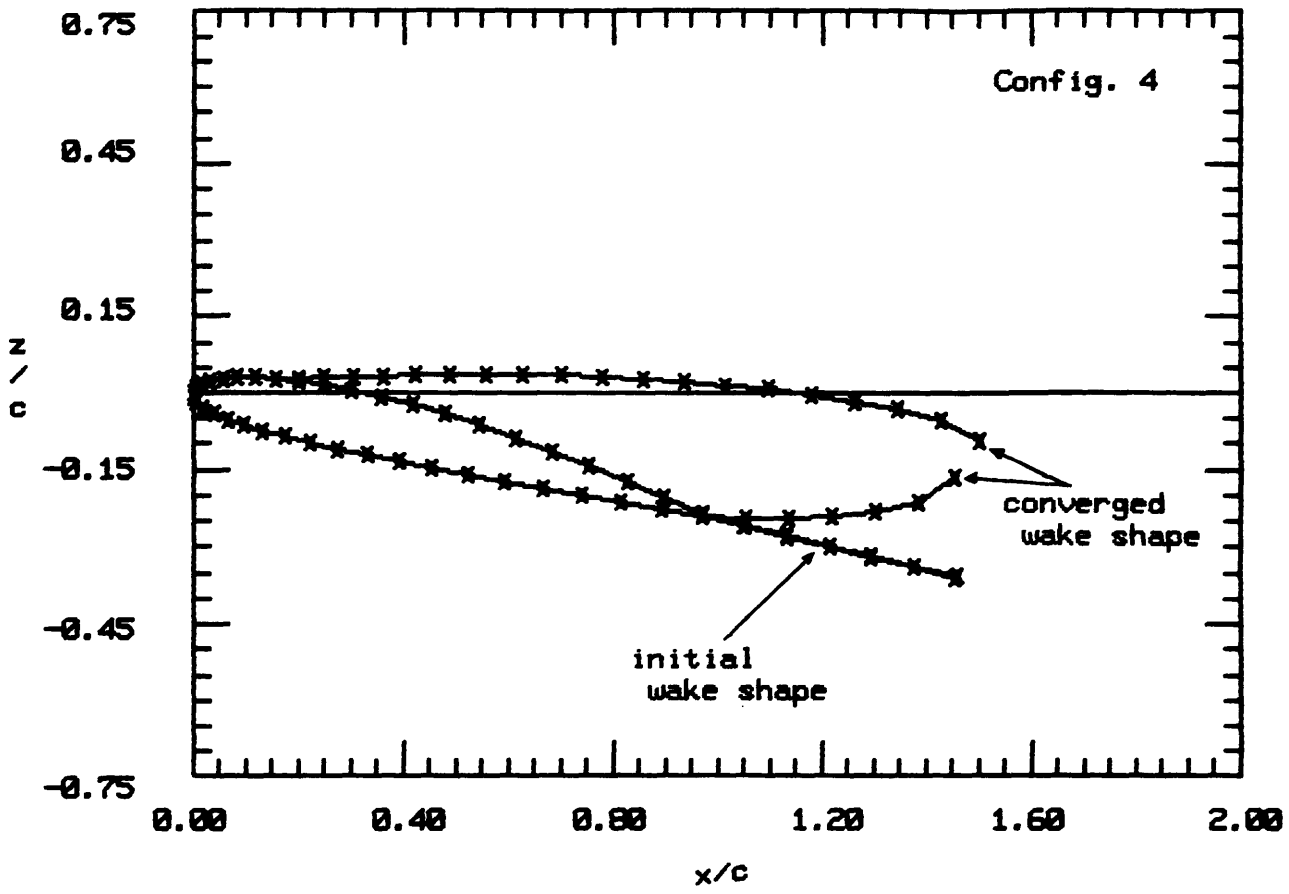


Figure 6-38 Initial and converged wake geometry at alpha = 13.8 deg

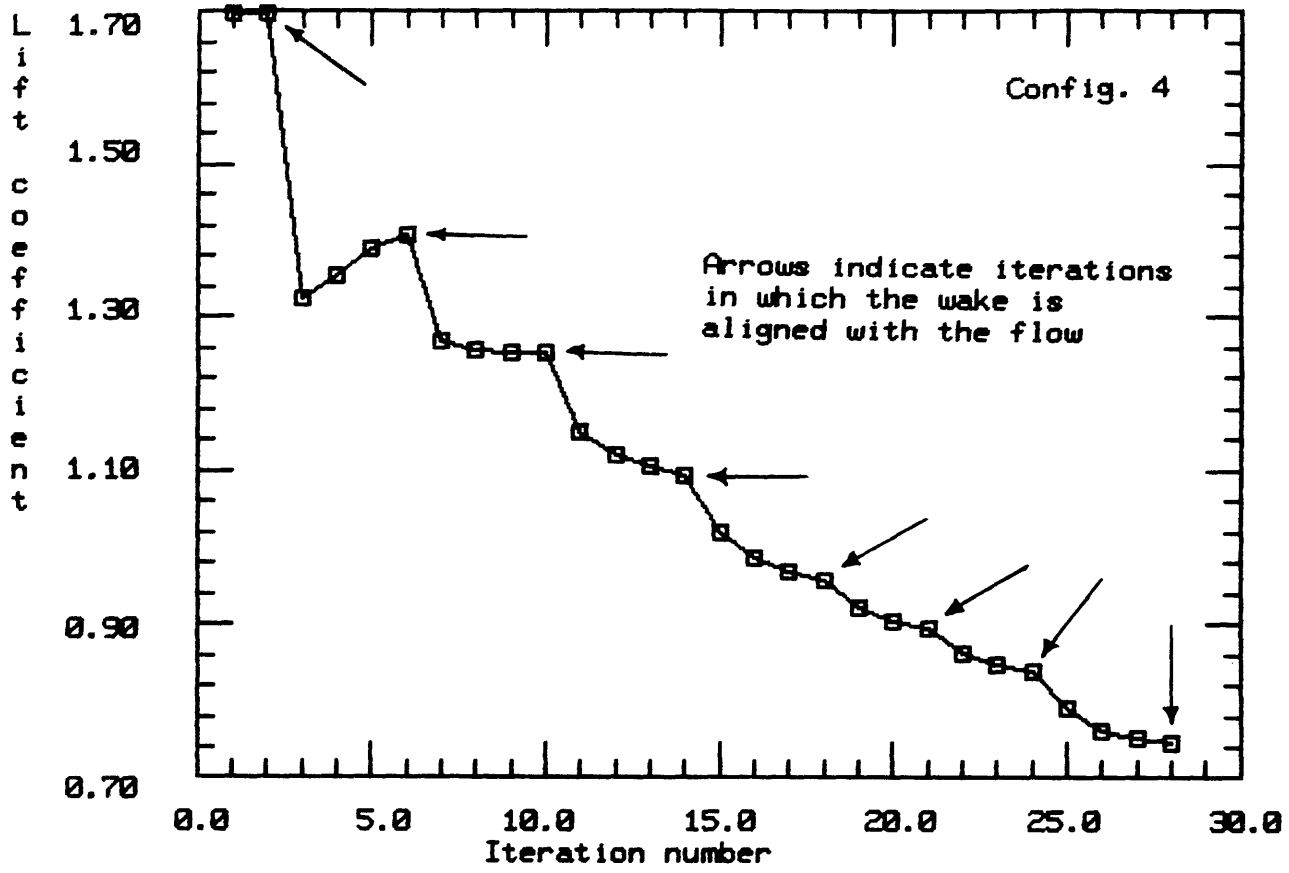


Figure 6-39 Convergence history for wing 4 at alpha = 13.8 deg

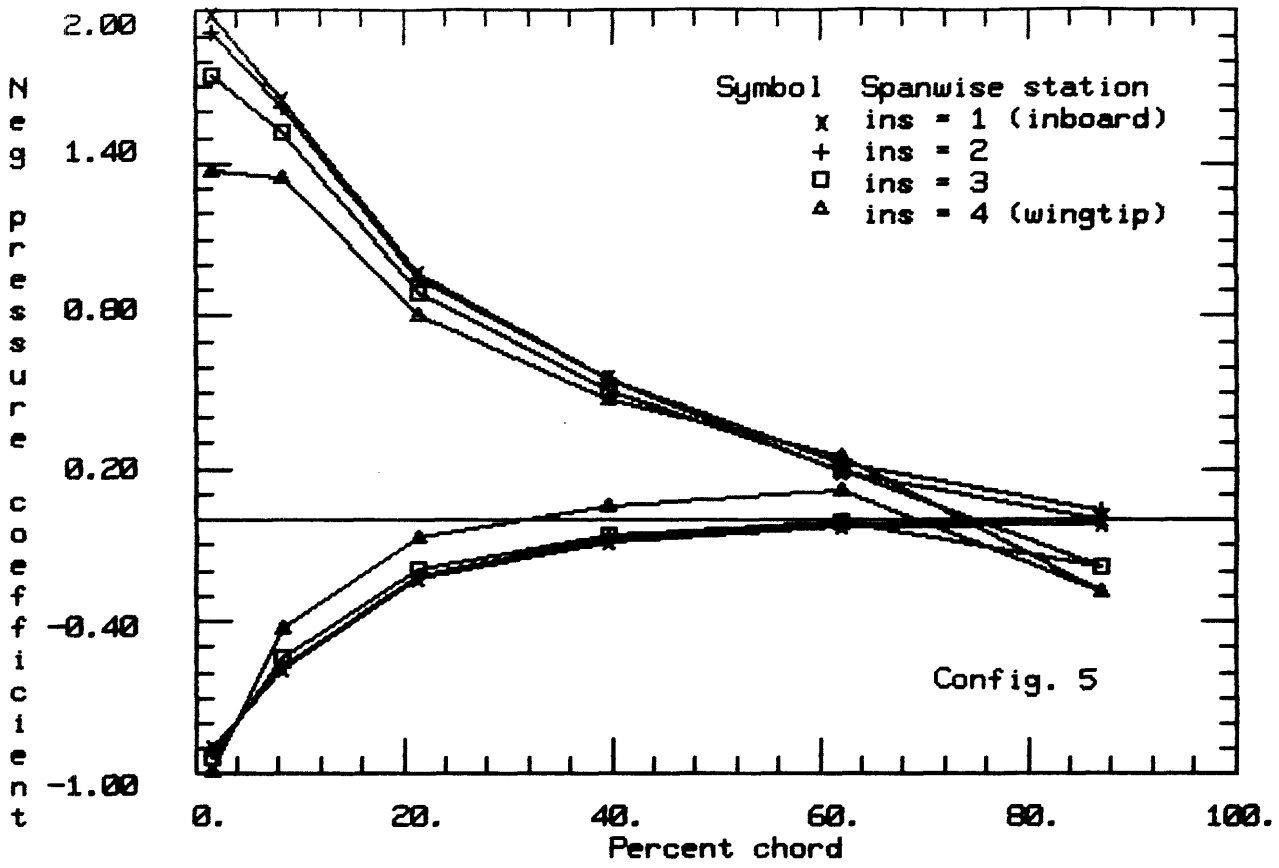


Figure 6-40 Wing pressure distribution in tunnel

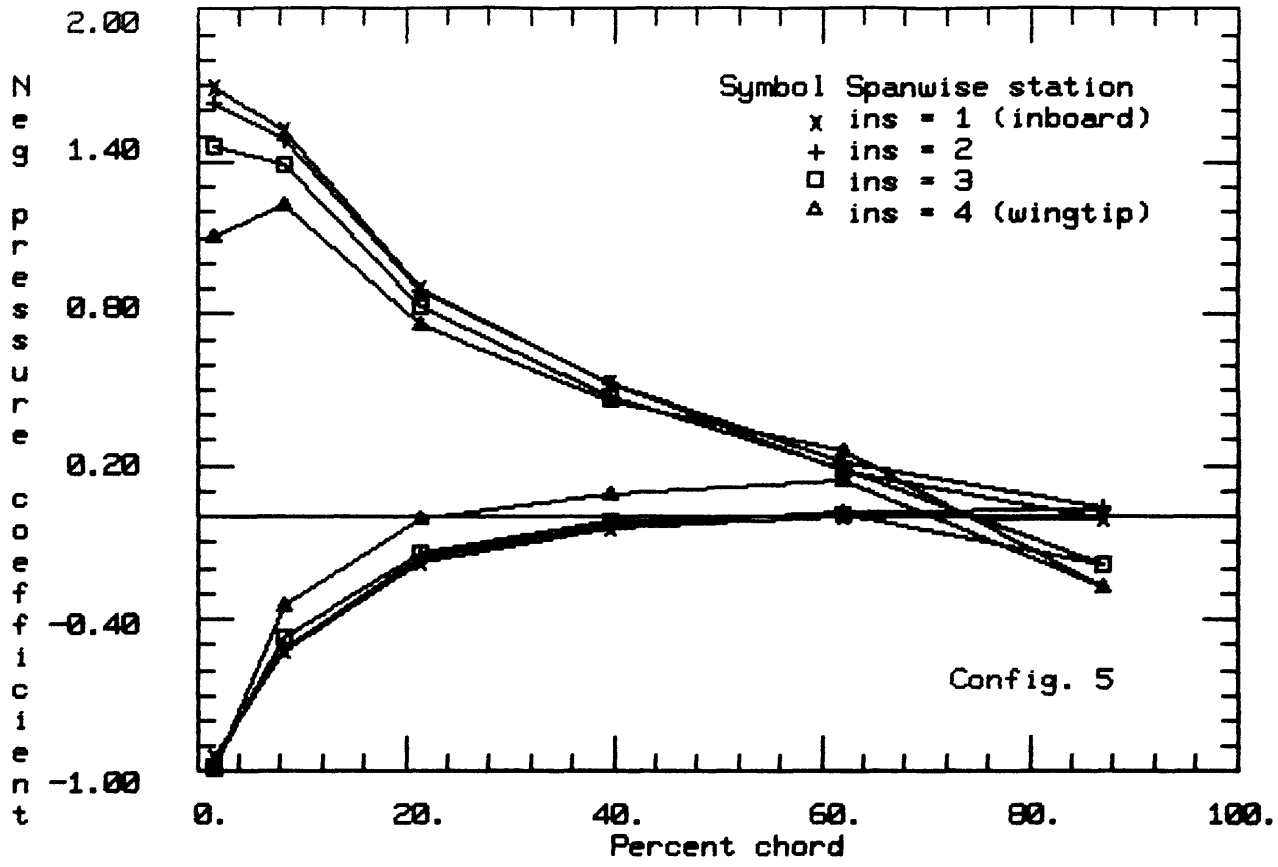


Figure 6-41 Wing pressure distribution in free air

REFERENCES

1) Whitnah, A.M. and Hillje, E.R., "Space Shuttle Wind Tunnel Testing Program Summary," NASA Reference Publication 1125, November 1984 Lyndon B. Johnson Space Center, Houston, Texas.

2) Rae, W.H.,jr and Pope, A., Low-Speed Wind Tunnel Testing, John Wiley & Sons, New York, 1984.

3) Joppa, R.G., "Wind Tunnel Interference Factors for High-Lift Wings in Closed Wind Tunnels," NASA CR-2191, February 1973.

4) Hackett, J.E. and Wilsden, O.J., "Determination of Low Speed Wake Blockage Corrrections via Tunnel Wall Static Pressure Measurements", AGARD-CP-174, Paper 23, October 1975.

5) Moses, D.F., "Wind Tunnel Wall Corrections Deduced by Iterating from Measured Wall Static Pressure," AIAA Journal, December 1983. pp. 206-231.

6) Sears, W.R., "Self Correcting Wind Tunnels," Aeronautical Quarterly, February/March 1974, pp. 80-89.

7) Davis, S.S., "Generalized Adaptive Wall Wind Tunnels," AIAA Paper 85-0225, January 1985.

8) Davis, S.S., "Applications of Adaptive-Wall Wind Tunnels," Journal of Aircraft, Vol. 23, February 1986, pp. 158-160.

9) Hess, J.L. and Smith, A.M.O., "Calculation of Potential Flow About Arbitrary Bodies," Progress in Aerospace Sciences, Vol. 8, p. 1-138 (Kuchemann ed.), Pergamon Press, 1966.

10) Mani, K.K., "A Multiple Separation Model for Multielement Airfoils," AIAA Paper 83-1844, July 1983.

11) Henderson, M.L., "Two-dimensional Separated Wake Modeling and its Use to Predict Maximum Section Lift Coefficient," AIAA 78-156.

12) Maskew, B. and Dvorak, F.A., "The prediction of Clmax Using a Separated Flow Model," American Helicopter Soc., April 1978.

13) Moran et. al., "Analysis of Two-Dimensional Incompressible Flows by a Subsurface Panel Method," AIAA Journal, vol. 18, pp. 526-533. 1980.

14) Olechowski, D.W., "Panel Methods for Wind Tunnel Wall Interference at Large Incidence with Separation." S.M. Thesis, Department of Aeronautics and Astronautics, M.I.T., May 1983.

15) Tennison, J., "An Interactive Boundary Layer/ Separated Flow Panel Method for Wind Tunnel Wall Corrections." S.M. Thesis, Department of Aeronautics and Astronautics, M.I.T., August 1985.

16) Bowcutt, K.G., "The Use of Panel Methods for the Development of Low-Subsonic Wall Interference and Blockage Corrections," M. S. Thesis, Department of Aerospace Engineering, University of Maryland, March 1984.

17) Bowcutt, K.G., "The Use of Panel Methods for the Development of Low-Subsonic Wall Interference and Blockage Corrections," AIAA-85-0159, January 1985.

18) Abbot, I.H. and VonDoenhoff, A.E., Theory of Wing Sections, Dover, New York, 1959.

19) Cebeci, T. and Bradshaw, P., "Momentum Transfer in Boundary Layers," McGrawhill/Hemisphere, Washington, D.C. 1977.

20) Thwaites, B., "Approximate Calculation of the Laminar Boundary Layer," Aeronautical Quarterly, vol. 1, November 1949 pp. 245-280.

21) Moran, J., "An Introduction to Theoretical and Computational Aerodynamics." John Wiley & Sons, New York, 1984.

22) Stratford B.S., "The Prediction of separation of the Turbulent Boundary Layer," Journal of Fluid Mechanics, vol. 5, pt. 1, 1959.

23) Kellogg, O.D., Foundations of Potential Theory, New York, Dover Publishing Co., 1953.

24) Bois, P., Tables of Indefinite Integrals, New York, Dover Publishing Co., 1961.

25) Strang, G., Linear Algebra and Its Applications, Academic Press Inc., 1980.

26) Schlichting, H., Truckenbrodt, E. and Ramm, H.J., Aerodynamics of the Airplane, New York, McGraw-Hill Inc., 1979.

27) Milne-Thomson, L.M., Theoretical Aerodynamics, New York, Dover Publishing Co., 1973.

28) Curle, N., "A Two-Parameter Method for Calculating the Two-Dimensional Incompressible Laminar Boundary Layer," Journal of the Royal Aeronautical Society, Vol. 71, 1967.