NASA Technical Memorandum 108852

Parametric Study on Laminar Flow for Finite Wings at Supersonic Speeds

Joseph Avila Garcia, Ames Research Center, Moffett Field, California

December 1994

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035-1000

TABLE OF CONTENTS

LIST OF FIGURES

PARAMETRIC STUDY ON LAMINAR FLOW FOR FINITE WINGS AT SUPERSONIC SPEEDS

Joseph Avila Garcia

Ames Research Center

SUMMARY

Laminar flow control has been identified as a key element in the development of the next generation of High Speed Transports. Extending the amount of laminar flow over an aircraft will increase range, payload, and altitude capabilities as well as lower fuel requirements, skin temperature, and therefore the overall cost. A parametric study to predict the extent of laminar flow for finite wings at supersonic speeds was conducted using a computational fluid dynamics (CFD) code coupled with a boundary layer stability code. The parameters investigated in this study were Reynolds number, angle of attack, and sweep. The results showed that an increase in angle of attack for specific Reynolds numbers can actually delay transition. Therefore, higher lift capability, caused by the increased angle of attack, as well as a reduction in viscous drag, due to the delay in transition, can be expected simultaneously. This results in larger payload and range.

INTRODUCTION

Background

Laminar flow control– Increasing the extent of laminar flow is equivalent to delaying boundary-layer transition. This delay in transition or control of laminar flow is obtained by passive, active, or reactive techniques (ref. 1). Passive techniques, also known as natural laminar flow (NLF) control, are categorized as those means of altering the boundary-layer flow through normal aerodynamic control parameters; for example, pressure-gradient, wall shaping, sweep, angle of attack, and Reynolds number.

Active techniques are categorized as those means of altering the flow through outside applied means; for example, wall suction, heat transfer.

A third form of flow control is reactive flow control. Reactive flow control is the process by which out-of-phase disturbances are artificially introduced into the boundary layer to cancel those disturbances already present, thus stabilizing the flow and delaying transition. Some reactive controls include periodic heating/cooling and wall motion. However, this method of laminar flow control is complex and, to date, is more of a theoretical method.

The underlying principle of these techniques, as one expert puts it, is "The realization that transition is the eventual stage in a process that involves amplification of disturbances in the boundary layer" (ref. 1).

Prediction of boundary-layer transition is an area which requires reliable methods and must be sensitive to any control parameter that alters the mean flow. These parameters include the active, passive, and reactive flow controls mentioned above.

Transition– The transition process is composed of several physical processes as described in figure 1.1 (ref. 1). The transition process begins by introducing external disturbances into the boundary layer through a viscous process known as receptivity (ref. 2). Some of these external disturbances include freestream vorticity, surface roughness, vibrations, and sound. Identifying and defining the initialization of these external disturbances, for a given problem, is the basis for the prediction of transition and creates an initial boundary-value problem. The initial disturbance is a function of the type of flow in consideration as well as its environment, and therefore is not usually known (ref. 1).

The disturbances in the boundary layer eventually enter the critical layer and then amplify. For low amplitude disturbances, the amplification can be modeled by linear stability theory. The normal modes responsible for the amplification of these disturbances in the boundary-layer flow are Tollmein–Schlichting (viscous) waves (or TS waves), Rayleigh (inflectional) waves (i.e., instabilities due to crossflow or high Mach numbers), and Görtler vortices for curved streamlines (ref. 1).

Once the amplifications are large enough, nonlinearity sets in through secondary and tertiary instabilities and the flow becomes transitional (ref. 1). It should be noted that the nonlinear portion of the flow is small compared to the linear region and therefore can still often be approximated by linear stability theory for preliminary designs.

One thing that must be avoided in all laminar flow studies is the introduction of high levels of initial nonlinear disturbances, which cause a bypass of the linear disturbance regime and yield an almost instantaneous transition. An example of such a nonlinear transition is attachment-line contamination, and is commonly found in swept wings due to the high crossflow at the wing leading-edge caused by turbulent flow from the fuselage.

Previous Work

Laminar flow control began in the 1930s with studies which investigated methods of natural laminar flow (NLF) control, specifically pressure gradient flows. This research led to the development of the NACA 6-series airfoils in the 1940s. Natural laminar flow research was later halted in the 1950s by the development of high speed jet engine aircraft. These jet aircraft reached transonic/supersonic speeds and required the wing to be swept to obtain lower local mach numbers and maintain reasonable aircraft performance (ref. 3). The effect of sweeping the wing introduced a three-dimensional crossflow instability that eliminated the ability to maintain laminar flow through current existing means. The sweepback and highly favorable pressure gradient near the leading-edge of the wing induces a boundary-layer crossflow. The sweep and adverse pressure gradient near the

trailing-edge likewise induces crossflow instabilities on the trailing-edge portion of the wing. Unlike the more common viscous two-dimensional TS instabilities, which are damped when a favorable pressure gradient is applied, the three-dimensional crossflow inflectional instabilities are amplified when pressure gradients exist (ref. 4). Therefore, by reducing the presence of pressure gradient flows over the wing, these crossflow instabilities can be reduced. One method of accomplishing this is by using NLF airfoils which produce low pressure-gradient flows.

Natural laminar-flow control research was then replaced by attempts to actively control boundary-layer transition, more commonly known as laminar flow control (LFC). These types of controls are categorized as active flow control, which began with flow suction on swept wings. The use of suction on the wing thins the boundary layer, lowering the effective Reynolds number, and moves the crossflow boundary-layer profile closer to the high viscous wall region, damping out crossflow instability, thus extending laminar flow (ref. 5). Work in this area peaked in the 1960s with the flight test of the X-21A. The X-21A's work showed the basic feasibility of extending LFC through active flow techniques at Reynolds numbers as high as 30 million (ref. 6).

Further development of the current research in LFC was delayed for a period of about ten years due to the decreased necessity to improve aircraft fuel efficiency caused by the abundance of low cost fuel resource and the high cost of designing such capabilities. It was not until the 1970s that interest in LFC research was recaptured and has continued to the present day.

The need for more fuel efficient aircraft has forced aircraft designers to consider fuel efficiency a top requirement. A major factor affecting fuel efficiency is turbulent skin friction drag. Advancements in aircraft skin material manufacturing processes to include strength and smoothness, as well as advancements in supercomputers and computing methods to analyze boundary-layer stability for transition prediction, have made laminar-flow control a more realistic method of improving aircraft fuel efficiency.

Turbulent skin friction drag is reduced by extending the amount of laminar flow over an aircraft. Until recently, most studies on laminar flow have been in the subsonic flow region. Work done in this subsonic realm has shown that turbulent skin friction drag can contribute as much as 50 percent of the total aircraft drag (ref. 7). Studies on typical Supersonic Transports (SSTs) have shown significant potential to increase the cruise lift-to-drag ratio by increasing the extent of laminar flow (refs. 8 and 9). Another benefit of laminar flow at supersonic speeds includes aerodynamic heating reduction, which allows for more skin/structure material options and, therefore, decreased aircraft gross weight and increased range/payload capability.

Current Work

A parametric study is being conducted as an effort to numerically predict the extent of natural laminar flow (NLF) on finite swept wings at supersonic speeds. This study is one part of the High Speed Research Program (HSRP) underway at NASA to gain an understanding of the technical requirements for supersonic laminar flow control (SLFC).

As mentioned previously, by extending laminar flow over the skin of an aircraft, there is a significant decrease in the turbulent skin friction which, in turn, decreases the total drag force on the aircraft's body. Furthermore, extending laminar flow at supersonic speeds will also significantly decrease the surface temperatures allowing for a more optimum selection of skin material.

By understanding the nature of supersonic laminar flow and the ability to control it, the following benefits can be expected in future High Speed Research (HSR) aircraft designs: increased range, increased payload, decreased fuel requirement, increased options for skin material, decreased initial cost, and decreased operating cost.

The parameters that are being addressed in this study are Reynolds number, angle of attack, and leading-edge wing sweep. These parameters were analyzed through the use of an advanced computational fluid dynamics (CFD) flow solver, specifically the Ames Research Center's three-dimensional compressible Navier–Stokes (CNS) flow solver (ref. 10). From the CNS code, pressure coefficients (Cp) are obtained for the various cases. These Cp's are then used to compute the boundary-layer profiles through the use of the "Kaups and Cebeci" compressible boundary-layer code (WING) (ref. 11). Finally, the boundary-layer parameters are fed into a three-dimensional compressible boundary-layer stability code (COSAL) to predict transition (ref. 12).

The parametric study consists of a Reynolds number study, an angle-of-attack study, and a leading-edge sweep study. The Reynolds number study addresses the Reynolds numbers of 6.34 million and 12.68 million at an angle of attack of 0 deg and leading-edge sweep of 45 deg. The angle-of-attack study addresses the angles of attack of 0, 5, and 10 deg at the two Reynolds number values and leading-edge sweep of 45 deg. Finally, the sweep study addresses the leading-edge sweeps of 45 and 60 deg at the lower Reynolds number and angle of attack of 0 deg. This yields a total of seven cases for the three studies. The above process was substantially automated through a procedure that was developed by the work conducted under this study. This automation procedure yields a three-dimensional graphical measure of the extent of laminar flow by predicting the transition location of laminar to turbulent flow.

GOVERNING EQUATIONS

Mean Flow

The physics of the flow in consideration can be described by the fundamental equations governing viscous fluid flow. These fundamental equations are based upon the universal laws of conservation of mass, momentum, and energy. These conservation laws are used to formulate the time-dependent, nondimensional Navier–Stokes equations in Cartesian coordinates (X, Y, Z) as given in the following vector form:

$$\frac{\partial \mathbf{Q}}{\partial t}+\frac{\partial \mathbf{E}}{\partial x}+\frac{\partial \mathbf{F}}{\partial y}+\frac{\partial \mathbf{G}}{\partial z}=\frac{\partial \mathbf{E}_v}{\partial x}+\frac{\partial \mathbf{F}_v}{\partial y}+\frac{\partial \mathbf{G}_v}{\partial z} \tag{2.1}$$

where the conserved quantity vector, **Q**, and the Euler flux vectors, **E**, **F**, **G**, are:

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(e+p) \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(e+p) \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(e+p) \end{bmatrix}$$

and the viscous flux vectors E_v, F_v, G_v, are:

$$\mathbf{E}_v = \mathrm{Re}^{-1} \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{yx} \\ \tau_{zx} \\ \beta_x \end{bmatrix}, \quad \mathbf{F}_v = \mathrm{Re}^{-1} \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{zy} \\ \beta_y \end{bmatrix}, \quad \mathbf{G}_v = \mathrm{Re}^{-1} \begin{bmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ \beta_z \end{bmatrix} \tag{2.2}$$

with

$$\begin{aligned}
\tau_{xx} &= \lambda(u_x + v_y + w_z) + 2\mu u_x \\
\tau_{yy} &= \lambda(u_x + v_y + w_z) + 2\mu v_y \\
\tau_{zz} &= \lambda(u_x + v_y + w_z) + 2\mu w_z \\
\tau_{xy} &= \tau_{yx} = \mu(u_y + v_x) \\
\tau_{xz} &= \tau_{zx} = \mu(u_z + w_x) \\
\tau_{yz} &= \tau_{zy} = \mu(v_z + w_y) \\
\beta_x &= \gamma\kappa\,\mathrm{Pr}^{-1}\,\partial_x e_\mathrm{I} + u\tau_{xx} + v\tau_{xy} + w\tau_{xz} \\
\beta_y &= \gamma\kappa\,\mathrm{Pr}^{-1}\,\partial_y e_\mathrm{I} + u\tau_{yx} + v\tau_{yy} + w\tau_{yz} \\
\beta_z &= \gamma\kappa\,\mathrm{Pr}^{-1}\,\partial_z e_\mathrm{I} + u\tau_{zx} + v\tau_{zy} + w\tau_{zz} \\
e_\mathrm{I} &= e/\rho - 0.5(u^2 + v^2 + w^2) \\
p &= (\gamma - 1)[e - 0.5\rho(u^2 + v^2 + w^2)]
\end{aligned} \tag{2.3}$$

The variables are nondimensionalized by dividing the spatial coordinates (x, y, z) by a reference length, L, the velocity components by the freestream speed of sound, a_∞, the density and viscosity by the corresponding freestream values, and the total energy per unit volume, e, by $(\rho a^2)_\infty$. A

Newtonian fluid is assumed with coefficient of bulk viscosity λ obtained from Stokes' hypothesis $\lambda = -2/3\mu$. It should be noted that "γ"is the ratio of specific heats, "κ" is the coefficient of thermal conductivity, "Re" is the Reynolds number, and "Pr" is the Prandtl number.

Coordinate transformation– To solve the governing equations, it is necessary to transform these equations into a generalized body-conforming, curvilinear coordinate system (ref. 10) as shown in figure 2.1. This allows the development of an efficient numerical algorithm, independent of body geometry, with a simplified application of the boundary conditions. This transformation maps the grid points in a one-to-one correspondence with the physical points, resulting in a grid with unit-volume cells everywhere (fig. 2.1). The general form of the transformation is expressed as:

$$(x,y,z) \rightarrow (\xi,\eta,\zeta)$$

$$\begin{aligned} \xi &= \xi(x,y,z) \\ \eta &= \eta(x,y,z) \\ \zeta &= \zeta(x,y,z) \end{aligned} \tag{2.4}$$

The chain rule of partial differentiation is applied to these transformation equations as follows:

$$\begin{aligned} \frac{\partial}{\partial x} &= x_x \frac{\partial}{\partial x} + h_x \frac{\partial}{\partial h} + z_x \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} &= x_y \frac{\partial}{\partial x} + h_y \frac{\partial}{\partial h} + z_y \frac{\partial}{\partial z} \\ \frac{\partial}{\partial z} &= x_z \frac{\partial}{\partial x} + h_z \frac{\partial}{\partial h} + z_z \frac{\partial}{\partial z} \end{aligned} \tag{2.5}$$

where the metric terms (ξ_x, η_x, ζ_x, ξ_y, η_y, ζ_y, ξ_z, η_z, ζ_z) appearing in equation 2.5 can be determined from the following matrix differential expressions:

$$\begin{bmatrix} \partial_\xi \\ \partial_\eta \\ \partial_\zeta \end{bmatrix} = \begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} \tag{2.6}$$

$$\begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{bmatrix} \begin{bmatrix} \partial_\xi \\ \partial_\eta \\ \partial_\zeta \end{bmatrix} \tag{2.7}$$

Therefore

$$\begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{bmatrix}^{-1}$$

$$= J\begin{bmatrix} y_\eta z_\zeta - y_\zeta z_\eta & -(x_\eta z_\zeta - x_\zeta z_\eta) & x_\eta y_\zeta - x_\zeta y_\eta \\ -(y_\xi z_\zeta - y_\zeta z_\xi) & x_\xi z_\zeta - x_\zeta z_\xi & -(x_\xi y_\zeta - x_\zeta y_\xi) \\ y_\xi z_\eta - y_\eta z_\xi & -(x_\xi z_\eta - x_\eta z_\xi) & x_\xi y_\eta - x_\eta y_\xi \end{bmatrix} \tag{2.8}$$

and the metric terms are represented as follows:

$$\begin{aligned} \xi_x &= J(y_\eta z_\zeta - y_\zeta z_\eta) \\ \xi_y &= J(z_\eta x_\zeta - x_\eta z_\zeta) \\ \xi_z &= J(x_\eta y_\zeta - x_\zeta y_\eta) \\ \eta_x &= J(z_\xi y_\zeta - y_\xi z_\zeta) \\ \eta_y &= J(x_\xi z_\zeta - x_\zeta z_\xi) \\ \eta_z &= J(y_\xi x_\zeta - x_\xi y_\zeta) \\ \zeta_x &= J(y_\xi z_\eta - y_\eta z_\xi) \\ \zeta_y &= J(x_\eta z_\xi - x_\xi z_\eta) \\ \zeta_z &= J(x_\xi y_\eta - x_\eta y_\xi) \end{aligned} \tag{2.9}$$

where J is the determinant of the Jacobian of the transformation

$$J = \frac{\partial(\xi,\eta,\zeta)}{\partial(x,y,z)} = \begin{vmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{vmatrix} \tag{2.10}$$

which can also be written as

$$J = 1/J^{-1} = 1/\frac{\partial(\xi,\eta,\zeta)}{\partial(x,y,z)} = 1/\begin{vmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{vmatrix} \tag{2.11}$$

$$= 1/[x_\xi(y_\eta z_\zeta - y_\zeta z_\eta) - x_\eta(y_\xi z_\zeta - y_\zeta z_\xi) + x_\zeta(y_\xi z_\eta - y_\eta z_\xi)]$$

Applying this transformation to the Navier–Stokes equations 2.1 gives

$$\frac{\partial \hat{\mathbf{Q}}}{\partial t} + \frac{\partial \hat{\mathbf{E}}}{\partial \xi} + \frac{\partial \hat{\mathbf{F}}}{\partial \eta} + \frac{\partial \hat{\mathbf{G}}}{\partial \zeta} = \left(\frac{\partial \hat{\mathbf{E}}_v}{\partial \xi} + \frac{\partial \hat{\mathbf{F}}_v}{\partial \eta} + \frac{\partial \hat{\mathbf{G}}_v}{\partial \zeta} \right) \quad \text{or}$$

$$\frac{\partial \hat{\mathbf{Q}}}{\partial t} + \frac{\partial}{\partial \xi}\left(\hat{\mathbf{E}} - \hat{\mathbf{E}}_v\right) + \frac{\partial}{\partial \eta}\left(\hat{\mathbf{F}} - \hat{\mathbf{F}}_v\right) + \frac{\partial}{\partial \zeta}\left(\hat{\mathbf{G}} - \hat{\mathbf{G}}_v\right) = 0 \tag{2.12}$$

where

$$\hat{\mathbf{Q}} = J^{-1}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix}, \qquad \hat{\mathbf{E}} = J^{-1}\begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ \rho w U + \xi_z p \\ (e+p)U \end{bmatrix}$$

$$\hat{\mathbf{F}} = J^{-1}\begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ \rho w V + \eta_z p \\ (e+p)V \end{bmatrix}, \qquad \hat{\mathbf{G}} = J^{-1}\begin{bmatrix} \rho W \\ \rho u W + \zeta_x p \\ \rho v W + \zeta_y p \\ \rho w W + \zeta_z p \\ (e+p)W \end{bmatrix}$$

$$\hat{\mathbf{E}}_v = \frac{J^{-1}}{Re}\begin{bmatrix} 0 \\ \xi_x\tau_{xx} + \xi_y\tau_{xy} + \xi_z\tau_{xz} \\ \xi_x\tau_{yx} + \xi_y\tau_{yy} + \xi_z\tau_{yz} \\ \xi_x\tau_{zx} + \xi_y\tau_{zy} + \xi_z\tau_{zz} \\ \xi_x\beta_x + \xi_y\beta_y + \xi_z\beta_z \end{bmatrix}, \quad \hat{\mathbf{F}}_v = \frac{J^{-1}}{Re}\begin{bmatrix} 0 \\ \eta_x\tau_{xx} + \eta_y\tau_{xy} + \eta_z\tau_{xz} \\ \eta_x\tau_{yx} + \eta_y\tau_{yy} + \eta_z\tau_{yz} \\ \eta_x\tau_{zx} + \eta_y\tau_{zy} + \eta_z\tau_{zz} \\ \eta_x\beta_x + \eta_y\beta_y + \eta_z\beta_z \end{bmatrix}$$

$$\hat{\mathbf{G}}_v = \frac{J^{-1}}{Re}\begin{bmatrix} 0 \\ \zeta_x\tau_{xx} + \zeta_y\tau_{xy} + \zeta_z\tau_{xz} \\ \zeta_x\tau_{yx} + \zeta_y\tau_{yy} + \zeta_z\tau_{yz} \\ \zeta_x\tau_{zx} + \zeta_y\tau_{zy} + \zeta_z\tau_{zz} \\ \zeta_x\beta_x + \zeta_y\beta_y + \zeta_z\beta_z \end{bmatrix} \tag{2.13}$$

where the components of the shear-stress tensor and heat-flux vector were given in equation 2.3 and the contra-variant velocity components (U, V, W) are

$$\begin{aligned} U &= \xi_x u + \xi_y v + \xi_z w \\ V &= \eta_x u + \eta_y v + \eta_z w \\ W &= \zeta_x u + \zeta_y v + \zeta_z w \end{aligned} \tag{2.14}$$

Thin-layer approximation– Large amounts of CPU time are necessary to solve the time-dependent three-dimensional Navier–Stokes equations, particularly for flow about complex geometries. To alleviate some of this large CPU requirement, a thin-layer approximation is applied to the governing equations. This thin-layer approximation is applicable to the present study involving only high Reynolds number flows, where the boundary layer is thin and the effects of viscosity are concentrated near the rigid boundaries. It should be noted that the thin-layer approximation requires that the body surface be mapped to a coordinate surface (for the present study $\xi = \xi_{min}$) and that clustering be normal to this surface. The resulting grid has fine grid spacing in the body-normal direction and much coarser spacing along the body. Therefore, the viscous terms in the body-normal direction are preserved and those viscous terms in the stream and spanwise direction are neglected. This approximation yields the following final form of the governing mean flow equations:

$$\frac{\partial \hat{\mathbf{Q}}}{\partial t} + \frac{\partial \hat{\mathbf{E}}}{\partial \xi} + \frac{\partial \hat{\mathbf{F}}}{\partial \eta} + \frac{\partial \hat{\mathbf{G}}}{\partial \zeta} = \frac{1}{Re}\left(\frac{\partial \hat{\mathbf{S}}}{\partial \zeta}\right) \tag{2.15}$$

where

$$\hat{\mathbf{S}} = J^{-1}\begin{bmatrix} 0 \\ \mu(\zeta_x^2+\zeta_y^2+\zeta_z^2)u_\zeta + \frac{\mu}{3}(\zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta)\zeta_x \\ \mu(\zeta_x^2+\zeta_y^2+\zeta_z^2)v_\zeta + \frac{\mu}{3}(\zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta)\zeta_y \\ \mu(\zeta_x^2+\zeta_y^2+\zeta_z^2)w_\zeta + \frac{\mu}{3}(\zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta)\zeta_z \\ \left\{ \begin{array}{l} (\zeta_x^2+\zeta_y^2+\zeta_z^2)\left[0.5\mu(u^2+v^2+w^2)\zeta + \dfrac{\mu}{\Pr(\gamma-1)}(a^2)\zeta\right] \\ \qquad + \dfrac{\mu}{3}(\zeta_x u + \zeta_y v + \zeta_z w)(\zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta) \end{array} \right\} \end{bmatrix}$$

and $\hat{\mathbf{Q}}$, $\hat{\mathbf{E}}$, $\hat{\mathbf{F}}$, and $\hat{\mathbf{G}}$ are given by equation 2.13.

Boundary-Layer Equations

Due to the extensive about of CPU time required to obtain an accurate boundary-layer solution, the Navier–Stokes mean flow solution was used to provide only the pressure distribution over the wing surface. This pressure distribution was then supplied to a boundary-layer code to provide the boundary-layer profiles needed to predict transition. The boundary-layer code WING was used. This boundary-layer code uses a conical flow approximation for the flow over a finite swept wing and assumes a polar coordinate system as shown in figure 2.2 (ref. 11). This conical flow assumption is valid for pressure isobars along constant percent chord lines for wings of trapezoidal planform. It should be noted that this assumption is not valid near the tip or root of the wing due to the strong pressure gradients created in these locations

The governing boundary-layer equations for the three-dimensional compressible laminar flow, with the above conical flow assumption ($\partial p/\partial r \equiv 0$), are given by the fundamental continuity, momentum, and energy equations and are expressed as:

Continuity equation:

$$\frac{\partial}{\partial r}(\rho r u) + \frac{\partial}{\partial \theta}(\rho w) + \frac{\partial}{\partial z}(\rho r v) = 0 \tag{2.16}$$

r-momentum equation:

$$\rho u\frac{\partial u}{\partial r} + \rho\frac{w}{r}\frac{\partial u}{\partial \theta} + \rho v\frac{\partial u}{\partial z} - \rho\frac{w^2}{r} = \frac{\partial}{\partial z}\left(\mu\frac{\partial u}{\partial z}\right) \tag{2.17}$$

θ-momentum equation:

$$\rho u\frac{\partial w}{\partial r}+\rho\frac{w}{r}\frac{\partial w}{\partial \theta}+\rho v\frac{\partial w}{\partial z}-\rho\frac{uw}{r}=-\frac{1}{r}\frac{\partial p}{\partial \theta}+\frac{\partial}{\partial z}\left(\mu\frac{\partial w}{\partial z}\right) \tag{2.18}$$

Energy equation:

$$\rho u\frac{\partial H}{\partial r}+\rho\frac{w}{r}\frac{\partial H}{\partial \theta}+\rho v\frac{\partial H}{\partial z}=\frac{\partial}{\partial z}\left[\frac{\mu}{Pr}\frac{\partial H}{\partial z}+\mu\left(1-\frac{1}{Pr}\right)\frac{\partial}{\partial z}\left(\frac{u^2+w^2}{2}\right)\right] \tag{2.19}$$

The following boundary conditions are then applied:

$$\text{at } y=0;\ u=0,\ v=v_w,\ w=0,\ \left(\frac{\partial H}{\partial y}\right)_w=0 \text{ (at the wall)}$$

$$\text{at } y\to\partial;\ u\to u_e,\ H\to H_e,\ w\to w_e \text{ (at the boundary-layer edge)}$$

where y is the distance normal to the wall, and the subscript w indicates the boundary-layer quantities at the wall. The symbol ∂ represents the boundary-layer thickness, and the subscript e is used to denote boundary-layer edge quantities.

Furthermore, u is the velocity component in the radial (r) direction, w is the velocity component normal to the radial direction, and v is the velocity component in the body-normal direction (fig. 2.2).

Finally, it should be noted that air is treated as a perfect gas, Sutherland's law is used for μ and the Prandtl number (Pr) is assumed constant.

Linear Stability Equations

The Compressible Stability AnaLysis (COSAL) code is used to analyze the stability of the three-dimensional boundary layer (ref. 12) in order to predict transition. COSAL determines the stability of the three-dimensional compressible Navier–Stokes equations using small-disturbance stability theory (ref. 13). Note, that the following derivation of the linear stability equation for the compressible three-dimensional flow will begin by deriving the incompressible flow (ρ = constant) condition for simplicity. The derivation will be completed with the derivation of the compressible stability equations.

Incompressible stability equations– The three-dimensional viscous incompressible flow is expressed by the following nonlinear Navier–Stokes equations:

$$\frac{\partial \mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\mathbf{u}=-\frac{1}{\rho}\nabla p+\nu\nabla^2\mathbf{u} \tag{2.20}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2.21}$$

The fluid motion is then decomposed into a steady flow and an instantaneous disturbance as follows:

$$\mathbf{u}(x,y,x,t) = \mathbf{U}(x,y,z) + \tilde{\mathbf{u}}(x,y,z,t) \tag{2.22}$$

$$\mathbf{p}(x,y,x,t) = \mathbf{P}(x,y,z) + \tilde{\mathbf{p}}(x,y,z,t) \tag{2.23}$$

where, **U** and **P** are the mean flow velocities and pressures respectively in the x, y, z directions. The x, y, z Cartesian coordinates are oriented so that x and z are the streamwise and spanwise directions, respectively, and y is the body-normal direction. These disturbances are substituted into equations 2.20 and 2.21. The basic terms of the original nonlinear Navier–Stokes equations are then subtracted away, and higher powers and products of the perturbation terms, being very small, are neglected. Finally, dynamic similitude is applied where all lengths are scaled by a reference length l, velocities by a reference velocity u_e, density by ρ_e, pressure by $\rho_e u^2_e$, and time by l/u_e, yielding the following linearized disturbance equations:

$$\frac{\partial \tilde{\mathbf{u}}}{\partial t} + \mathbf{U} \cdot \nabla \tilde{\mathbf{u}} + \tilde{\mathbf{u}} \cdot \nabla \mathbf{U} = -\nabla \tilde{\mathbf{p}} + \frac{\mathbf{1}}{\mathbf{R}} \nabla^2 \tilde{\mathbf{u}} \tag{2.24}$$

$$\nabla \cdot \tilde{\mathbf{u}} = 0 \tag{2.25}$$

where R is a characteristic Reynolds number defined as:

$$R = \frac{\rho_e y_e l}{\mu_e}$$

Furthermore, a "quasiparallel" flow is assumed, which implies that the mean flow is only a function of the body-normal coordinate "y" for a given point along the body. This means the velocity only varies in the y direction and not in the x or z direction. This assumption is applicable to boundary-layer flows since, at high Reynolds numbers, the flow gradients in the streamwise (x) or spanwise (z) direction are much smaller than in the body-normal (y) direction. The quasiparallel flow assumption can therefore be represented as follows:

$$\mathbf{U} = U(U(y), 0, W(y)) \tag{2.26}$$

where U(y) and W(y) are the velocity components in the x and z directions, respectively.

The linear disturbance equations are now homogeneous, separable partial differential equations (PDEs), and the following normal mode solution applies:

$$\begin{Bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \\ \tilde{p} \end{Bmatrix} = \begin{Bmatrix} \hat{u}(y) \\ \hat{v}(y) \\ \hat{w}(y) \\ \hat{p}(y) \end{Bmatrix} \exp[i(\alpha x + \beta z - \omega t)] \tag{2.27}$$

where α and β are the x and z components of the disturbance wave vector, $\bar{k}$, as shown in figure 2.3 (ref. 1), and $\hat{u}, \hat{v}, \hat{w}, \hat{p}$ are the complex eigenfunctions that determine the structure of the disturbance for a given frequency (ω).

Substituting equations 2.26 and 2.27 into the linearized Navier–Stokes disturbance equations 2.24 and 2.25 yields the following set of ordinary differential equations (ODEs):

$$i(\alpha U + \beta W - \omega)\hat{u} + \frac{dU}{dy}\hat{v} = -i\alpha\hat{p} + \frac{1}{R}\left[\frac{d^2\hat{u}}{d^2y} - \hat{u}(\alpha^2 + \beta^2)\right] \tag{2.28}$$

$$i(\alpha U + \beta W - w)\hat{v} = -\frac{d\hat{p}}{dy} + \frac{1}{R}\left[\frac{d^2\hat{v}}{d^2y} - \hat{v}(\alpha^2 + \beta^2)\right] \tag{2.29}$$

$$i(\alpha U + \beta W - \omega)\hat{w} + \frac{dW}{dy}\hat{v} = -i\beta\hat{p} + \frac{1}{R}\left[\frac{d^2\hat{w}}{d^2y} - \hat{w}\left(\alpha^2 + \beta^2\right)\right] \tag{2.30}$$

$$i\alpha\hat{u} + i\beta\hat{w} + \frac{d\hat{v}}{dy} = 0 \tag{2.31}$$

Next, the following boundary conditions are applied:

at $y = 0$ (wall); $\quad \hat{u}(0) = \hat{v}(0) = \hat{w}(0) = 0$

as $y \to \infty$ (freestream); $\quad \hat{u}(y) \to 0, \quad \hat{v}(y) \to 0, \quad \hat{w}(y) \to 0$

Note that the boundary conditions and equations 2.28 through 2.31 are homogeneous; therefore an eigenvalue problem exists and a solution exists for only a certain combination of α, β, and ω. This solution can be expressed by the following dispersion relation:

$$\omega = \omega(\alpha, \beta) \tag{2.32}$$

where α, β, ω are all complex.

Now there exists the following six arbitrary real parameters:

$$(\alpha_r, \alpha_i, \beta_r, \beta_i, \omega_r, \omega_i)$$

which then form an eigenvalue problem.

Compressible stability equations– The three-dimensional viscous compressible flow stability equations are an extension of the above derived incompressible equations.

The fluid motion is decomposed into a steady flow and an instantaneous disturbance, as was done for the incompressible nonlinear Navier–Stokes equations, as follows:

$$u(x,y,x,t) = U(x,y,z) + \tilde{u}(x,y,z,t) \tag{2.33}$$

$$p(x,y,x,t) = P(x,y,z) + \tilde{p}(x,y,z,t) \tag{2.34}$$

$$\tau(x,y,x,t) = T(x,y,z) + \tilde{\tau}(x,y,z,t) \tag{2.35}$$

Note that the temperature term τ was added to take into account the compressibility effects.

The Cartesian coordinate system x, y, z is used again in which the y-axis is normal to the solid body and x, z are parallel to it. The term u, represents the x, y, z components of the instantaneous velocity, respectively, and p and τ are the instantaneous pressure and temperature. Next, equations 2.33 through 2.35 are substituted into the nonlinear compressible Navier–Stokes equations. The resulting equation is linearized by subtracting away the basic terms of the original nonlinear Navier–Stokes equations and neglecting higher powers and products of the perturbation terms. Finally, assuming the basic flow is locally parallel as was done above in the quasiparallel flow assumption of equation 2.26, the linearized compressible Navier–Stokes equations become separable, permitting the following normal mode solution:

$$\begin{Bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \\ \tilde{p} \\ \tilde{t} \end{Bmatrix} = \begin{Bmatrix} \hat{u}(y) \\ \hat{v}(y) \\ \hat{w}(y) \\ \hat{p}(y) \\ \hat{\tau}(y) \end{Bmatrix} \exp[i(\alpha x + \beta z - \omega t)] \tag{2.36}$$

Here, the quantities with tildas denote complex disturbance amplitudes.

Substituting equations 2.26 and 2.36 into the linearized compressible Navier–Stokes equations yields the following system of ordinary differential equations:

$$(AD^2 + BD + C)\bar{\phi} = 0 \tag{2.37}$$

where D represents "d/dy" and $\bar{\phi}$ is the vector defined by

$$\bar{\phi} = \begin{Bmatrix} \alpha\tilde{u} + \beta\tilde{w} \\ \tilde{v} \\ \tilde{p} \\ \tilde{\tau} \\ \alpha\tilde{w} - \beta\tilde{u} \end{Bmatrix} \tag{2.38}$$

and A, B, C are 5×5 matrices given by

$$A = \begin{Bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{Bmatrix}$$

$$B = \begin{Bmatrix} \frac{1}{\mu_o}\frac{d\mu_o}{dT_o}T'_o & i(\lambda-1)/\lambda & 0 & \frac{2(\gamma-1)M^2\sigma(\alpha U'_o + \beta W'_o)}{(\alpha^2+\beta^2)} & 0 \\ i(\lambda-1)(\alpha^2+\beta^2) & \frac{1}{\mu_o}\frac{d\mu_o}{dT_o}T' & 1 & 0 & 0 \\ 0 & -\frac{R}{\mu_o\lambda} & 0 & 0 & 0 \\ \frac{1}{\mu_o}\frac{d\mu_o}{dT_o}(\alpha U'_o + \beta W'_o) & 0 & 0 & \frac{2}{\mu_o}\frac{d\mu_o}{dT_o}T'_o & \frac{1}{\mu_o}\frac{d\mu_o}{dT_o}(\alpha W'_o - \beta U'_o) \\ 0 & 0 & 0 & \frac{2(\gamma-1)M^2\sigma(\alpha W'_o - \beta U'_o)}{(\alpha^2+\beta^2)} & \frac{1}{\mu_o}\frac{d\mu_o}{dT_o}T'_o \end{Bmatrix}$$

$$
C = \begin{Bmatrix}
\left\{\begin{matrix} \frac{-iR}{\mu_o T_o}(\alpha U_o + \beta W_o \\ + \omega) - \lambda(\alpha^2 + \beta^2)] \end{matrix}\right\} & \left\{ i\frac{\chi}{\lambda\mu_o}\frac{d\mu_o}{dT_o}T' \right\}_o & 0 & 0 & 0 \\
\left\{\begin{matrix} \frac{-R}{\mu_o T_o}(\alpha U_o' + \beta W_o') \\ + \frac{i}{\mu_o}\frac{d\mu_o}{dT_o}T_o'(\alpha^2 + \beta^2) \end{matrix}\right\} & \left\{\begin{matrix} -[\frac{iR}{\mu_o T_o}(\alpha U_o + \beta W_o \\ - \omega) + (\alpha^2 + \beta^2)/\lambda] \end{matrix}\right\} & \left\{ -\frac{T_o'}{T_o} \right\} & \left\{\begin{matrix} -[iR\sigma T_o'/T_o\mu - 2i(\gamma \\ -1)M^2\sigma(\alpha U_o' + \beta W_o')] \end{matrix}\right\} & \left\{ \frac{-R}{\mu_o T_o}(\alpha W_o' + \beta U_o') \right\} \\
\left\{ -\frac{iR}{\mu_o}(\alpha^2 + \beta^2) \right\} & 0 & \left\{\begin{matrix} i\gamma M^2 \\ (\alpha U_o \\ +\beta W_o \\ +\omega) \end{matrix}\right\} & \left\{\begin{matrix} iR\sigma/\mu_o T_o(\gamma - 1)M^2 \\ (\alpha U_o + \beta W_o - \omega) \end{matrix}\right\} & 0 \\
\left\{\begin{matrix} \frac{1}{\mu_o}[(\alpha U_o' + \beta W_o')\frac{d^2\mu_o}{dT^2{}_o} \\ T_o' + (\alpha U_o'' + \beta W_o'')\frac{d\mu_o}{dT_o}] \end{matrix}\right\} & \left\{ \frac{i}{\lambda\mu_o}\frac{d\mu_o}{dT_o}(\alpha U_o' + \beta W_o') \right\} & 0 & \left\{\begin{matrix} -[\frac{iR\sigma}{\mu_o T_o}(\alpha U_o + \beta W_o \\ -\omega) + (\alpha^2 + \beta^2) - (\gamma - \\ 1)M^2\frac{1}{\mu_o}(\frac{d\mu_o}{dT_o}(U_o'^2 \\ + W_o'^2) - \frac{d^2\mu_o}{dT^2{}_o}(T_o')^2 \\ -\frac{d\mu_o}{dT_o}T_o''] \end{matrix}\right\} & \left\{\begin{matrix} \frac{1}{\mu_o}[\frac{d\mu_o}{dT_o}(\alpha W_o' - \beta U_o') \\ +\frac{d\mu_o}{dT_o}(\alpha W_o'' - \beta U_o'')] \end{matrix}\right\} \\
0 & 0 & 0 & 0 & \left\{\begin{matrix} -[\frac{iR\sigma}{\mu_o T_o}(\alpha U_o + \beta W_o \\ - \omega) + (\alpha^2 + \beta^2)] \end{matrix}\right\}
\end{Bmatrix}
\tag{2.39}
$$

The boundary conditions for equation 2.37 are

$$
y = 0;\ \phi_1 = \phi_2 = \phi_4 = \phi_5 = 0
$$

$$
y \to \infty;\ \phi_1 = \phi_2 = \phi_4 = o_5 \to 0 \tag{2.40}
$$

The above boundary conditions and equations 2.37–2.39 represent an eigenvalue problem as was found for the incompressible derivation represented earlier. This eigenvalue problem can also be expressed by the dispersion relation of equation 2.23 which relates the wavenumber vector components α, β with the complex frequency ω. Also, note that again there exists the six arbitrary real parameters

$$(\alpha_r, \alpha_i, \beta_r, \beta_i, \omega_r, \omega_i)$$

Solution of the eigenvalue problem– The eigenvalue problem can be solved by specifying four of the six parameters mentioned above and finding the other two parameters by using equations 2.28–2.31 for the incompressible flow, or 2.37–2.39 for compressible flow. In order to solve the eigenvalue a temporal stability theory is used which assumes that the disturbance grows or decays only in time (temporally) and not in space (spatially). Since α, β are the spatial parameters and ω is the temporal parameter (i.e., see eq. 2.36) of the disturbance, then α, β are assumed to be real and ω complex. Therefore, the disturbance amplification is represented by the complex component of the frequency (ω_i) and grows or decays as follows:

$$\omega_i > 0, \text{ grows}$$

$$\omega_i < 0, \text{ decays}$$

Then a disturbance level measurement (N-factor or N) is obtained for transition and is represented as follows:

$$N = \int_{S_c}^{S_t} \frac{\omega_i}{\mathrm{Re}(\bar{V}_g)} \, ds \tag{2.41}$$

where $\bar{V}_g$ is the group velocity (direction and speed of the wave energy) (fig. 2.3). Assuming a two-dimensional wave, the Gaster transformation (ref. 15) can be used to estimate the group velocity as

$$\bar{V}_g = \frac{\partial \omega}{\partial \alpha} \tag{2.42}$$

Note, $\omega = 2\pi f$ and $\alpha = f \cdot \lambda$ is the wave number. The group velocity yields the change in frequency (ω) from one location to another downstream location.

To compute the N-factor, the real frequency (f) and the disturbance wave length (λ) must be specified. The N-factor is then integrated along the curve tangent to the real part of the group velocity. Transition is then predicted at an N-factor of 8 to 10 based on comparison with empirical data from previous studies on swept wings (refs. 4, 16, and 17).

NUMERICAL METHODS

Mean Flow

There are two finite-difference scheme options in the compressible Navier–Stokes (CNS) code to solve the thin-layer Navier–Stokes equations. These finite-difference schemes are the implicit approximation factorization algorithm in delta form by Beam and Warming (ref. 18) and the diagonal implicit algorithm by Pulliam and Chaussee (ref. 19).

Implicit methods are used over explicit methods to avoid restrictive time-step stability conditions which occur when small grid spacing is required, as in the present study. This high resolution grid spacing requirement is needed to capture the boundary-layer viscous effects occurring near the wall in the present study. Unlike explicit methods that yield stiff problems and restrict the time step to very small values for stability, implicit methods generally avoid such stiffness problems and allow the use of a larger time step without loss of accuracy (ref. 19).

Beam–Warming block ADI algorithm– The Beam–Warming algorithm is first- or second-order accurate in time and second- or fourth-order accurate in space. The equations are spatially split or factored to reduce the process to a set of one-dimensional problems for each time iteration. The algorithm produces a 5×5 block tridiagonal system that must be inverted for each spatial dimension for each time step, due to the second-order central-difference operators being used. Further discussion of the accuracy and stability characteristics of this numerical scheme can be found in Beam and Warming (ref. 18). Based on linear analysis, the following numerical scheme is unconditionally stable in two dimensions but in actual use, time step limits are encountered because of the nonlinear nature of the equations. The algorithm in three-dimensions is unconditionally unstable, although through the use of artificial dissipation terms the stability is maintained.

Pulliam–Chaussee diagonal ADI algorithm– The second basic numerical algorithm used to solve the Navier–Stokes equations in the CNS code has been taken from the Pulliam–Steger ARC3D computer code (ref. 19). This algorithm is known as the Pulliam–Chaussee Diagonal ADI algorithm. This scheme uses a fourth-order-accurate smoothing operator on both the left- and right-hand sides. In this algorithm the flux Jacobians are diagonalized by special similarity transformations which greatly simplify the iteration process. For this algorithm only a set of scalar pentadiagonal matrices need be inverted for each time step, making this scheme several times less expensive than the Beam–Warming block scheme described above. The Pulliam–Chaussee diagonal algorithm was used for all mean flow computations in the present study.

Boundary-Layer Equations

The boundary-layer code WING uses the Keller box method to solve the boundary-layer equations 2.16–2.19. This method has been proven to be an accurate and efficient method to solve parabolic partial differential equations of this type, as found in references 20–23.

Boundary-Layer Stability Equations

The compressible boundary-layer stability equations 2.37 are solved by the COSAL code using a second-order finite-difference formulation (ref. 14). The code includes two eigenvalue search procedures. A global eigenvalue search procedure is used when no guess is available for the eigenvalues. A local eigenvalue search is used when a good guess for the eigenvalues is available; this is approximately 10 times faster than the global procedure (ref. 14).

COMPUTATIONAL GRID AND BOUNDARY CONDITIONS

Wing Grid Configurations

The computational grids used in this analysis were generated from an algebraic surface grid generation code named wing surface generator or WSG developed in this study. The airfoil ordinates, required by the above surface grid code, were obtained from a code developed to obtain ordinates for NACA 6- and 6A-series airfoils (ref. 24). This code produces airfoils of a given thickness, thickness distribution, or camber. These ordinates are then redistributed using either the interactive surface grid generation codes S3D (ref. 25) or visual grid (VG) (ref. 26). Once the desired airfoil section is acquired and the surface grid is generated, the three-dimensional grid is then generated through the use of the hyperbolic volume grid generator HYPGEN (ref. 27).

Wing surface generator (WSG)– The algebraic surface grid generation code mentioned above was developed to quickly generate various wing geometries. This code generates single-element wings with specified sweep and taper ratio for a given airfoil shape. Appendix A contains a set of instructions for program execution, appendix B contains a copy of the code, and appendix C has several required pre-processing codes. WSG was designed to allow the user a quick method of creating single-element wing surface grids. The following is a list of the inputs: taper ratio or aspect ratio; leading edge or quarter chord sweep; desired number of spanwise cuts on the wing; initial spacing in the spanwise direction (wing-tip spacing); final spacing in the spanwise direction (wing-root spacing); and airfoil ordinates file.

It should be noted that the process necessary to obtain the above-mentioned airfoil ordinate input file requires a few steps and is described in the flow chart of figure 4.1. For a detailed explanation of the process, refer to the instructions listed in appendix A.

The surface grid generation code requires only a few seconds execution time on an IRIS workstation. One feature of the code includes a check for negative trailing-edge sweep, which can be obtained when certain combinations of taper ratio, aspect ratio, and leading-edge sweep are chosen. The reason for this check is due to the fact that the boundary-layer code currently being used in the transition analysis cannot analyze swept-forward wing edges.

Finally, note that the algebraic surface grid generator uses the Vinokur stretching routine (ref. 28) to cluster points along the spanwise direction at the wing's wake, root, and tip sections.

Volume grid generator– The three-dimensional computational grids for the various wing geometries being studied are generated using a hyperbolic three-dimensional grid generation code HYPGEN (ref. 27). This code generates a three-dimensional volume grid over the generated single-block surface grids. HYPGEN accomplishes this by solving the three-dimensional hyperbolic grid generation equations consisting of two orthogonality relations and one cell volume check.

The cell volume check is one of two grid quality checks conducted by HYPGEN after a grid is generated. The cell volume check is a cell volume computation using tetrahedron decomposition, and checks the grid for any types of distortions. The second test is a Jacobian computation and uses a

finite volume algorithm, specifically the OVERFLOW flow solver algorithm (ref. 29). If a grid passes the two tests, it should run through the flow solver. Although, if any cell in the grid passes the second test and not the first test, the accuracy may be affected in those regions (ref. 27).

Boundary Conditions

The solid wall conditions are specified in CNS as no-slip and adiabatic. The outer boundary or far-field flow variables are set to free-stream flow conditions. A symmetry plane is used at the wing's root which eliminates effects due to the fuselage that could yield leading-edge flow contamination also known as spanwise turbulent contamination. This phenomenon was first discovered by Gray (ref. 30) in flight at the Royal Aircraft Establishment (RAE) in 1951 and is a nonlinear transition as was discussed earlier in the Introduction.

AUTOMATED STABILITY ANALYSIS

In order to conduct the following parametric study it was necessary to substantially automate the analysis process due to the extensive number of man-hours required to obtain a transition prediction. Once the automation portion of the study was completed it was necessary to perform validation. The F-16XL Ship1 flight test was used as a validation case.

The automated stability analysis process was created using a script that combined the three codes used in this study to obtain transition predictions, as illustrated in figure 5.1. A copy of the script can be found in appendix D. The automation process begins after a file is generated from the mean flow solution. This file contains the pressure distributions of the selected span stations for a specific wing geometry. The pressure distributions are supplied to the boundary-layer code (WING), one span station at a time, which computes the boundary-layer parameters and profiles. Next the boundary-layer outputs are supplied to the Compressible Stability AnaLysis code (COSAL) to measure the disturbances in the boundary-layer. Note that for each span station the stability analysis requires that the script run the stability code for a spectrum of frequencies between 0 and 40,000 Hz to determine the most unstable condition. This is accomplished by setting up a loop in the script to run a set of 23 input files with the required COSAL input for the spectrum of frequencies. Finally, an outer loop is required in the script to analyze the selected span stations.

The user time required for an average COSAL run is approximately 30 seconds and since the frequency scan requires 23 runs for each of the 8 selected span stations on the wing, a total average CPU time of 1.5 hours on a single processor Cray Y-MP is needed per case. The actual turnaround time for a typical job may run as long as 3 hours due to the added I/O time to run the boundary-layer code WING and other post-processing codes in the developed automation script. Also, it should be noted that the stability analysis must be run with 64-bit precision (e.g., Cray Y-MP) due to the needed accuracy of the eigenvalue search routine used in COSAL.

RESULTS AND DISCUSSION

As mentioned in section 1, extending laminar flow over the skin of an aircraft significantly decreases the skin friction which, in turn, decreases the total drag force on the aircraft. This drag reduction will allow for increased range/payload and decreased fuel requirements. Furthermore, extending laminar flow at supersonic speeds will also significantly decrease the surface temperatures, allowing for a more optimum selection of skin material.

The parameters addressed in the present study are Reynolds number, angle of attack, and leading-edge wing sweep. Since this study is being focused on High Speed Civil Transport (HSCT) type aircraft, the range of angle of attack is limited to 10 deg. The Reynolds number of 1.12 million per foot based on a Mach number of 1.5 and altitude of approximately 45,000 feet is used. The leading-edge sweeps consist of 45 and 60 deg.

Stability Automation Validation

In order to validate the automation process of section 5, the F-16XL wing was used. The results of the F-16XL wing transition validation case, using the newly developed automated stability process, compared well with the results previously obtained manually, as shown in figure 6.1. As a result of the automated process the number of man hours required to obtain a single three-dimensional transition front dropped from hours to a matter of minutes, and the overall turnaround time dropped from days to a matter of hours.

Reynolds Number Effects

Before a full parametric study was conducted, it was necessary to establish a baseline case that had a reasonable region of laminar flow. This was necessary so that the effects of changing the various parameters could be distinguished. To achieve a fair amount of laminar flow, maintain supersonic cruise conditions at 40,000 to 50,000 feet altitude, and achieve a free stream Mach number of 1.5, a Reynolds number of 1.27 million per foot was used. The Reynolds number was then varied by changing the root chord length. The results of the Reynolds number study showed that the extent of laminar flow was decreased as the local Reynolds number was increased. This is illustrated in figure 6.2 by the transition fronts of the chosen baseline wing for two root chord lengths, 5 and 10 feet. The light gray region signifies the portion of the wing where laminar flow is no longer predicted. The dark gray region represents laminar flow as predicted for disturbance levels (N-factors) in the boundary layer ranging from 0 to 8. The disturbance level of 8 is selected, indicated by the solid black line, as the critical transition N-factor for all of the following parametric studies. The critical disturbance level of 8 was chosen as a conservative value based on previous swept-wing transition prediction studies (refs. 4, 16, and 17). It should be noted that the transition results near the tip and root of the wing are not valid due to the conical flow assumption used in the boundary-layer code (WING). Tip effects also eliminate the potential for laminar flow near the wing tip region. The analysis was therefore only limited to the gray area shown in figure 6.3. Further investigation into the conical flow assumption showed that for this configuration the flow was not

truly conical, as can be seen in the pressure coefficient (Cp) plots for the two Reynolds number cases in figure 6.4. These pressure coefficient plots show the chordwise pressure distribution versus the normalized x/c locations for the eight span station locations computed in the boundary-layer stability analysis. Note that if the flow was truly conical the Cp distribution for each span station would basically be swept back and, when plotted normalized with the local chord length, they would all have the same Cp distribution. The Cp distribution results (fig. 6.4) show that from the mid-semispan (48 percent semispan) to the tip of the wing some conical flow does occur for approximately the first 20 percent chord. The Cp distributions also show that for the 33 percent semispan conical flow is only valid up to approximately 10 percent chord, and for 13 percent to 19 percent semispan the conical flow assumption is not valid at all. It should also be noted that at higher angles of attack the wing tip effect becomes more pronounced, further diminishing conical flow near the tip. Therefore, the wing root and tip regions will not be discussed further. This study will only include the mid-semispan (48 percent semispan) station of the wing.

Furthermore, the pressure distribution result (fig. 6.4) shows that there exists a strong favorable pressure gradient at the leading-edge of the wing and a strong adverse pressure gradient at the wing's trailing edge. As was mentioned earlier in the introduction, laminar flow transition studies have found that the three-dimensional crossflow instabilities are amplified in the presence of pressure gradient flows and that the more common two-dimensional TS instabilities are damped in the presence of favorable pressure gradient flows. Therefore, all boundary-layer stability calculations to determine transition will be conducted for crossflow instabilities and not TS instabilities.

In order to study the flow more thoroughly, boundary-layer profile plots were made for the two Reynolds number cases at 48 percent semispan. Since transition is found to occur before 20 percent chord, crossflow boundary-layer profiles were plotted from x/c of 0 percent to 21 percent, as shown in figure 6.5. The crossflow profiles reveal that the inflection point of the profile moves closer to the wall as the Reynolds number is decreased. Since the Reynolds number is varied by changing the root chord of the wing, the boundary-layer's normal distance from the wall (y) is nondimensionalized with the local boundary layer thickness (d). The crossflow profile at the x/c station of 10 percent is considered first. When the crossflow profile is now plotted as y/d versus the crossflow component, the results show that the two Reynolds number cases follow exactly the same trend (fig. 6.6). Also, figure 6.6 indicates that the inflection of the crossflow profiles occur at the same y/d for the two different Reynolds number cases However, a plot of y/d versus shear stress at the x/c of 10 percent for the two Reynolds number cases (fig. 6.7) reveals that, at a given y/d, the shear stresses in the boundary layer are lower for the higher Reynolds number case and higher for the lower Reynolds number case. Only the x/c station of 10 percent is shown to illustrate the relationship of the movement of the crossflow inflection point of figure 6.5 to the shear stress (fig. 6.7) due to Reynolds number effects. Finally, the crossflow boundary-layer profile results (figs. 6.5 and 6.6) show that changes in Reynolds number do not affect the magnitude of the maximum crossflow velocities.

Next, stability curves of the transition results at the 48 percent semispan station are shown in figure 6.8. This figure is a plot of chordwise x/c versus frequency for the Reynolds number study at the critical boundary-layer disturbance level (N-factor) of 8. Basically, this plot shows the x/c locations at which the given frequencies yield the disturbance level of 8, and it is defined that the x/c value where this disturbance level first occurs is where transition is predicted. For example, for the Reynolds number of 6.3 million, the curve indicates that the disturbance level of 8 first occurs at the

x/c value of approximately 12 percent for a frequency of 14,000 Hz. For the higher Reynolds number case of 12.7 million, the results show that the transition shifts forward to an x/c of approximately 3 percent and a frequency of 20,000 Hz. Therefore, from the results of the linear-stability theory's transition prediction and the boundary-layer profiles, its is revealed that a decrease in Reynolds number yields higher shear stresses in the boundary layer which act to damp out the crossflow instabilities and delay transition.

Angle-of-Attack Effects

Recall that the results in the wing tip and root regions are not valid, due to the conical flow assumption made in the boundary-layer solutions. Therefore, the boundary-layer and transition prediction results will only be addressed at the mid-semispan (48 percent semispan) station for all the following cases.

Boundary-layer stability curves of the transition analysis at 48 percent semispan are shown in figure 6.9. This figure consists of three curves which are plots of x/c versus frequency for the three angles of attack at the boundary-layer disturbance level of 8. The results for the angle-of-attack case of 0 deg indicate that the most unstable frequency is 14,000 Hz, and the earliest transition location occurs at an approximate x/c value of 12.25 percent. The 5 deg angle-of-attack case results show that transition moves back to approximately 18.5 percent chord at a critical frequency of 12,000 Hz. Although the 10 deg case shows that the transition only moves back to 15.75 percent chord at a critical frequency of 14,000 Hz. Therefore, this shows that for an increase in angle of attack, transition moves aft and that certain angles of attack produce more delay in transition than others.

In order to study why an increase in angle of attack revealed this trend in the delay of transition, a plot of the chordwise pressure distribution for the three angle-of-attack cases at 48 percent semispan is shown in figure 6.10. The result shows that as angle of attack is increased a stronger favorable pressure gradient at the leading-edge occurs for the first 5 percent chord and then a smaller favorable pressure gradient continues up to approximately 80 percent chord. The swept wing's three-dimensional crossflow is expected to further destabilize with increase in angle of attack, due to the presence of the stronger pressure gradients. Therefore, it is expected that the prediction in transition would move further forward. However, this trend does not occur.

Next, the surface flow patterns of the different angle-of-attack cases are shown (fig. 6.11) to study the flow characteristics. These patterns reveal a separation occurring near the trailing edge of the wing as the angle of attack increases to 10 deg. In order to better see how the flow pattern is affected near the leading-edge, where the flow on the wing first begins, a plot of the leading-edge flow at 48 percent semispan is shown in figure 6.12. The dashed lines indicate the flow trace over the upper wing surface, including the leading-edge point, and the solid lines indicate the flow trace over the lower wing surface. From this plot it is evident that the flow attachment point moves below the leading-edge on to the lower surface of the wing as angle of attack increases. It should also be noted that because the attachment point rotates below the leading-edge as angle of attack is increased, the crossflow velocities at the leading-edge location are reduced.

It was found from a previous parametric study on the leading-edge attachment line of the F-16XL (ref. 31) that the maximum crossflow velocity at a given wing location decreased as the angle of attack increased due to the rotation of the attachment point underneath the leading-edge. It was then expected that transition would also rotate forward. Although, the findings of this study show that the opposite trend occurs.

To further investigate the above findings, boundary-layer crossflow profile curves are displayed in figure 6.13 for the three angle-of-attack cases at approximately mid-semispan (48 percent semispan). The boundary-layer profile curves are plotted for x/c from 0 to 21 percent. Results of the crossflow profiles reveal that the crossflow velocity components are larger for the higher angle-of-attack cases near the leading-edge. However, further downstream the trend reverses and the lower angle-of-attack cases exhibit higher crossflow values. In order to better represent this trend a plot of the maximum crossflow "(W/U_∞)max" versus "x/c" for the different angles of attack is shown in figure 6.14. This plot shows that the maximum crossflows are larger for the higher angles of attack (5 and 10 deg) up to approximately 5 percent chord. After 5 percent chord the maximum crossflows for the 5 deg angle-of-attack case fall below the 0 deg angle-of-attack case. The 10 deg angle-of-attack case falls below the 0 deg angle-of-attack case at approximately 8 percent chord and remains below the 5 deg angle-of-attack case after 16 percent chord. Finally, the 0 deg angle-of-attack case slowly falls but remains above the two higher angle-of-attack cases after 5 percent chord.

In summary, increasing angle of attack shows that near the leading edge, the maximum crossflow is larger for the higher angles of attack, and downstream the lower angle-of-attack cases exhibit higher maximum crossflow. This translates into a 6.25 percent chord increase in laminar flow as the wing's angle of attack is increased to 5 deg and a 3.25 percent chord increase for the 10 deg case. This leads to the speculation that transition may be directly influenced by maximum crossflow in the boundary layer which is discussed further in the next section.

Reynolds Number Effects with Angle of Attack

The results of the angle of attack study show that maximum crossflow may directly influence the transition prediction. The results of the Reynolds number study show that a decrease in Reynolds number increases the shear stresses in the boundary layer (fig 6.5(b)) thereby damping out the crossflow instabilities which then delays the predicted transition location. The above findings indicate that maximum crossflow may have a major influence on the predicted transition location. To validate this possibility, another angle-of-attack study at the higher Reynolds number flow of 12.7 million was conducted in order to see how transition is affected with changes in angle of attack. This Reynolds number flow is chosen since the earlier results (fig. 6.8) show that transition occurs at nominally 3 percent chord for 0 deg angle of attack.

Crossflow boundary-layer profiles for the three angles of attack of 0, 5, and 10 deg at the higher Reynolds number of 12.7 million are shown in figure 6.15. The results from these crossflow profiles show that the maximum crossflow is larger for the higher angles of attack near the leading edge. Further downstream the maximum crossflow is larger for the lower angle-of-attack cases. These results are the same as in the previous lower Reynolds number angle-of-attack study (fig. 6.13). A

plot of maximum crossflow versus chordwise x/c location for this study (fig. 6.16) reveals almost exactly the same results as in the previous lower Reynolds number angle-of-attack study.

However, as in the Reynolds number study, the inflection points of the crossflow profiles for the higher Reynolds numbers (fig. 6.15) are further away from the viscous wall when compared to the earlier lower Reynolds number case (fig. 6.13). Therefore, this translates into lower shear stresses in the boundary layer for the higher Reynolds number case, which leads to a suspected transition further upstream near the leading edge. In this region, the crossflow results (fig. 6.16) show that maximum crossflows are higher at the higher angles of attack.

Transition is now predicted to move forward as the angle of attack is increased, as shown in the boundary-layer stability curves of figure 6.17. These results therefore validate that maximum crossflow has a direct influence on the predicted transition location.

Sweep Effects

In addition to investigating the effects of angle of attack, the effects of sweep were also studied. It was necessary to keep the wing's aspect ratio constant so that the comparison in sweep would not be misinterpreted by other changes in the wing's surface area or local chord. It was also necessary to avoid sweeping the wing into the Mach cone, which would cause shock waves and distort the flow. Due to the above requirements, it was necessary to shear the baseline clipped delta wing to obtain the 60 deg sweep as well as maintain the same aspect ratio and local chord lengths, as shown in figure 6.18.

Flow traces of the two different wing sweep cases in figure 6.19, show that the 60 deg-swept-wing case appears to have a flow separation from about 30 percent of the semispan all the way to the tip near the trailing edge of the wing. Although, as mentioned earlier, only the mid-semispan will be evaluated in detail.

The results of the crossflow effects due to sweep (fig. 6.20) show, at all x/c locations up to 21 percent except at 1 percent, that the crossflows for the 60 deg sweep are stronger than the 45 deg sweep. Furthermore, results of the maximum crossflow "(W/U_∞max" versus streamwise location "x/c" plot (fig. 6.21) shows that the maximum crossflow is slightly larger for the 45 deg sweep at 1 percent chord and then drops below that of the 60 deg sweep at 3 percent chord. The 60 deg sweep case maximum crossflows are larger after 2 percent chord and slightly fluctuate after 10 percent chord. Overall, these results show that maximum crossflow is larger for the higher 60 deg swept wing.

Next, stability curves of the transition results at 48 percent semispan are shown in figure 6.22. This is the same type of plot as the one discussed earlier in the angle-of-attack study. The results show that transition occurs at approximately an x/c of 12 percent and a frequency of 14,000 Hz for the 45 deg sweep case. For the 60 deg sweep case, transition is predicted to occur at an x/c of approximately 10 percent and a frequency of approximately 20,000 Hz. Transition moves forward approximately 2 percent of chord when the wing is swept from 45 to 60 deg. Therefore, these results

also show that maximum crossflow is a major influence on transition prediction as found in the angle-of-attack and Reynolds number studies above.

CONCLUSIONS AND RECOMMENDATIONS

A parametric study to predict the extent of laminar flow for finite wings at supersonic speeds was successfully completed using a computational fluid dynamics code coupled with a boundary-layer stability code. The study was conducted to gain understanding of the technical requirements in the area of supersonic laminar flow control (SLFC) to assist in the High Speed Research Program (HSRP) underway at NASA. The effects of Reynolds number, angle of attack, and sweep were investigated.

Conclusions

The results of automating the boundary-layer stability analysis has reduced the time required to predict the three-dimensional transition front location from hours to a matter of minutes. Furthermore, the automation has reduced the overall turnaround time for a transition front prediction from days to a matter of hours.

The results of the Reynolds number study show that a decrease in Reynolds number increases the amount of laminar flow over the wing and can be attributed to the effects of Reynolds number on crossflow instabilities. Essentially, the crossflow boundary-layer profile is moved closer to the high viscous wall region when Reynolds number is decreased, damping out the crossflow instabilities and therefore increasing the extend of laminar flow.

The results of the angle-of-attack study revealed that an increase in angle of attack moves the attachment point beneath the leading-edge of the wing and increases the maximum crossflow near the leading-edge. However, further downstream the maximum crossflow velocities are lower for the higher angles of attack.

The results of the combined effects of Reynolds number and angle of attack show that transition can actually be delayed with an increase in angle of attack for specific Reynolds numbers. This means of delaying transition was accomplished by decreasing the Reynolds number so that transition is delayed to the point where the maximum crossflow is lower for the higher angle of attack. The result is an increase in the laminar flow over the wing and therefore a reduction in the viscous drag on the wing. An advantage to this type of natural laminar flow (NLF) control is that the drag increase due to lift (caused by the increase in angle of attack) can partially be recovered by the viscous drag reduction due to the increase in the laminar flow over the wing. The results basically show that if maximum crossflow is decreased near the location where transition is predicted to occur, then transition can further be delayed. Finally, the results of the sweep study again show that maximum crossflow is a key to transition prediction. As the wing is swept back an increase in the crossflow occurs, increasing the crossflow instability and therefore allowing an earlier transition prediction.

Recommendations

In the future, a leading-edge shape study should be conducted to find the effects of bluntness at supersonic speeds on the extent of laminar flow.

A total drag calculation for the various flow conditions and swept-wing configurations should be computed. This would reveal the actual effect of Reynolds number, angle of attack, and sweep on total drag.

Investigation of the numerical methods being applied show that the two-dimensional boundary-layer code which uses a conical flow assumption is not truly valid for swept wings and should be replaced with a three-dimensional boundary-layer code. Furthermore, it is recommended that future research use the boundary-layer information from the Navier–Stokes solutions in place of the boundary-layer solutions.

Finally, the results of this Supersonic Natural Laminar Flow Study should be combined with active supersonic laminar-flow control methods in order to establish an optimum method of achieving supersonic laminar flow for future high speed aircraft design.

REFERENCES

1. Malik, M. R.: Stability Theory for Laminar Flow Control Design. Progress in Astronautics and Aeronautics, vol. 123, 1990, pp. 3–46.

2. Morkovin, M. V.: Critical Evaluation of Transition from Laminar to Turbulent Shear Layer with Emphasis on Hypersonically Traveling Bodies. AFFDL-TR-68-149 (available as NTIS AD-686178), 1969.

3. Bushnell, D. M.; and Tuttle, M. H.: Survey and Bibliography on Attainment of Laminar Flow Control in Air Using Pressure Gradient and Suction. NASA RP-1035, vol. I, Sept. 1979.

4. Srokowski, A. J.: Mass Flow Requirement for LFC Wing Design. AIAA Paper 77-1222, Aug. 1977.

5. Betchov, R.; and Criminale, W. O.: Stability of Parallel Flows. Academic Press, New York 1967.

6. Kosin, R. E.: Laminar Flow Control by Suction as Applied to the X-21 Airplane. J. Aircraft, vol. 2, Sept.–Oct. 1965, pp. 384–390.

7. Fischer, M. C.; and Ash, Robert L.: A General Review of Concepts for Reducing Skin Friction, Including Recommendations for Future Studies. NASA TM-X-2894, March 1974.

8. Beckwith, I. E.: Development of High Reynolds Number Quiet Tunnel for Transition Research. AIAA J., vol. 13, no. 3, March 1975, pp. 300–306.

9. Pfenninger, W.: USAF and Navy Sponsored Northrop LFC Research Between 1949 and 1967. Workshop on Laminar Flow Control, compiled by C. T. D'Aiutolo, NASA Langley Research Center, Apr. 6–7, 1976, pp. 14–50.

10. Kaynak, U.; Holst, T. L.; and Cantwell, B. J.: Computational of Transonic Separated Wing Flow Using an Euler/Navier–Stokes Zonal Approach. NASA TM-88311, July 1986.

11. Kaups, K.; and Cebeci, T.: Compressible Laminar Boundary Layers with Suction on Swept and Tapered Wings. J. Aircraft, vol. 14, no. 7, July 1977, pp. 661–667.

12. Malik, M. R.: COSAL: A Black-Box Compressible Stability Analysis Code for Transition Prediction in Three Dimensional Boundary Layers. High Technology Corp., NASA CR-165925, 1982.

13. White, F. M.: Viscous Fluid Flow. 2nd ed., Mc Graw-Hill Series in Mech. Engineering.

14. Malik, M. R.: Efficient Computation of the Stability of Three-Dimensional Compressible Boundary Layers. AIAA Paper 81-1277, Palo Alto, Calif., June 1981.

15. Gaster, M.: A Note on the Relation Between Temporally-Increasing and Spatially-Increasing Disturbances in Hydrodynamic Stability. J. Fluid Mechanics, vol. 14, Oct. 1962, pp. 222–224.

16. Hefner, J. N.; and Bushnell, D. M.: Application of Stability Theory to Laminar Flow Control. AIAA Paper 79-1493, July 1979.

17. Cebeci, T.; Chen, H. H.; and Arnal, D.: A Three-Dimensional Linear Stability Approach to Transition on Wings at Incidence. AGARD CP-438, Oct. 1988, pp. 17-1–17-13.

18. Beam, R.; and Warming, R. F.: An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation-Law Form. J. Comp. Physics, vol. 22, Sept. 1976, pp. 87–110.

19. Pulliam, T. H.: Euler and Thin-Layer Navier Stokes Codes: ARC2D, and ARC3D. Computational Fluid Dynamics Users' Workshop, The Univ. of Tenn. Space Inst., Tullahoma, Tenn., March 12–16, 1984

20. Keller, H. B.: A New Difference Scheme for Parabolic Problems. Numerical Solutions of Partial Differential Equations, vol. II, ed. by J. Bramble, Academic Press, New York, 1970.

21. Cebeci, T.; and Bradshaw, P.: Momentum Transfer in Boundary Layers. Hemisphere and McGraw Hill, 1977.

22. Keller, H. B.; and Cebeci, T.: Accurate Numerical Methods for Boundary Layers. II. Two-Dimensional Turbulent Flows. AIAA J., vol. 10, Sept. 1972, pp. 1197–1200.

23. Cebeci, T.; and Smith, A. M. O.: Analysis of Turbulent Boundary Layers. Academic Press, New York, 1974.

24. Ladson, Charles L.; and Brooks, Cuyler W., Jr.: Development of a Computer Program to Obtain Ordinates for NACA 6- and 6A-Series Airfoils. NASA Langley Research Center, June 25, 1974.

25. Luh, R. C.; Pierce, L. E.; and Yip, D.: Interactive Surface Grid Generation. AIAA Paper 91-0796, Reno, Nev., Jan. 1991.

26. Cordova, J. Q.: Visual Grid, A Software Package for Interactive Grid Generation. AIAA Paper 90-1607, Seattle, Wash., June 18, 1990.

27. Chan, W. M.; and Steger, J. L.: A Generalized Scheme For Three-Dimensional Hyperbolic Grid Generation. AIAA Paper 91-1588, Proceedings of the AIAA 10th Computational Fluid Dynamics Conference, Honolulu, Hawaii, 1991.

28. Vinokur, M.: On One-Dimensional Stretching Function for Finite-Difference Calculations. J. Computational Phys., vol. 50, no. 2, May 1983.

29. Buning, P. G.; Chan, W. M.; Renze, K. J.; Sondak, D.; Chiu, I. T.; and Slotnick, J. P.: OVERFLOW User's Manual, Version 1.6. NASA Ames Res. Ctr., Moffett Field, Calif., 1991.

30. Gray, W. E.: The Effect of Wing Sweep on Laminar Flow. RAE TM Aero 255, 1952.

31. Flores, J.; Tu, E. L.; Anderson, B.; and Landers, S.: A Parametric Study of the Leading Edge Attachment Line for the F-16XL. AIAA Paper 91-1621, Proceedings of the AIAA 22nd Fluid Dynamics, Plasma Dynamics and Laser Conference, Honolulu, Hawaii, June 24–26, 1991.

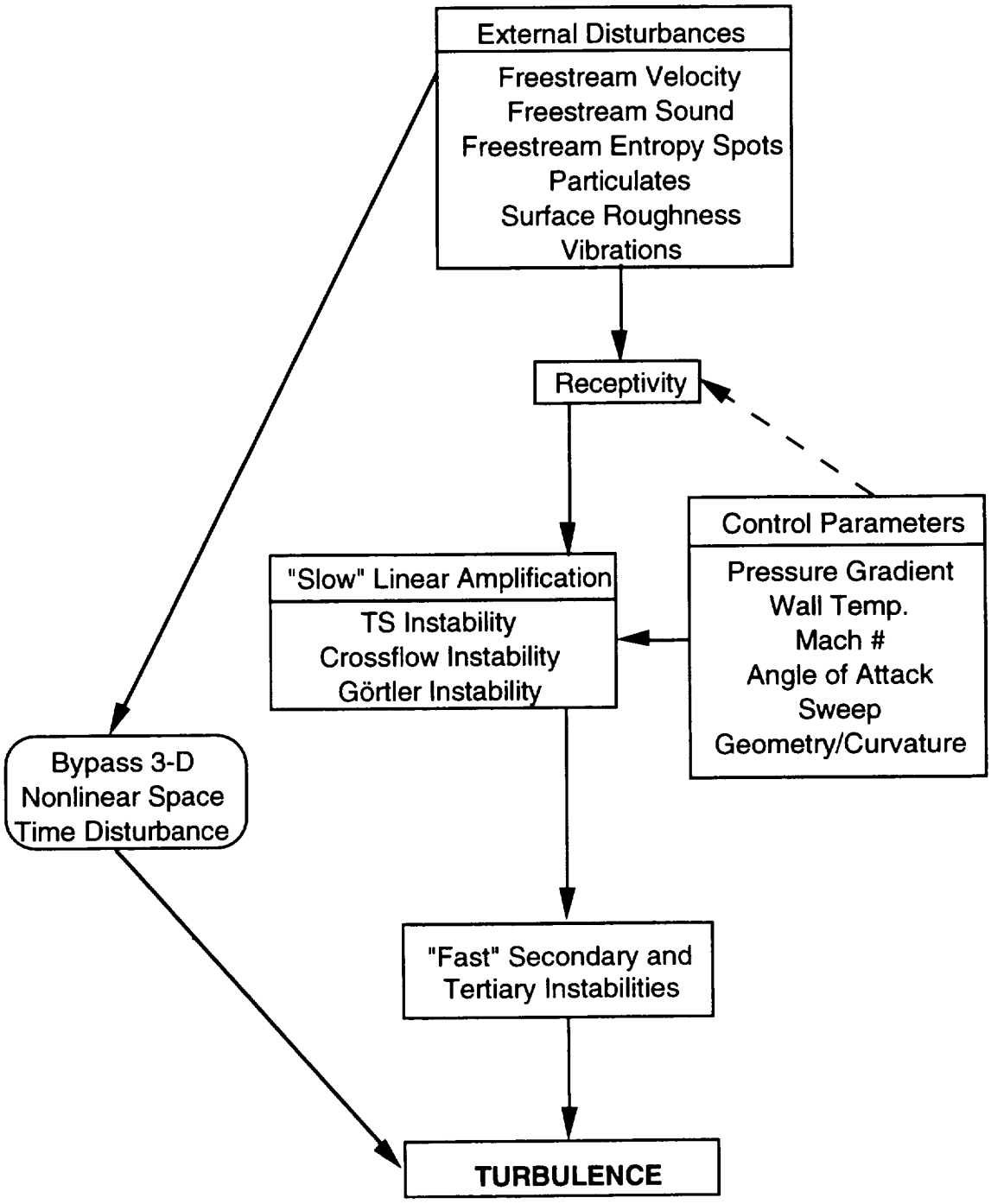


Figure 1.1. Transition flow chart.

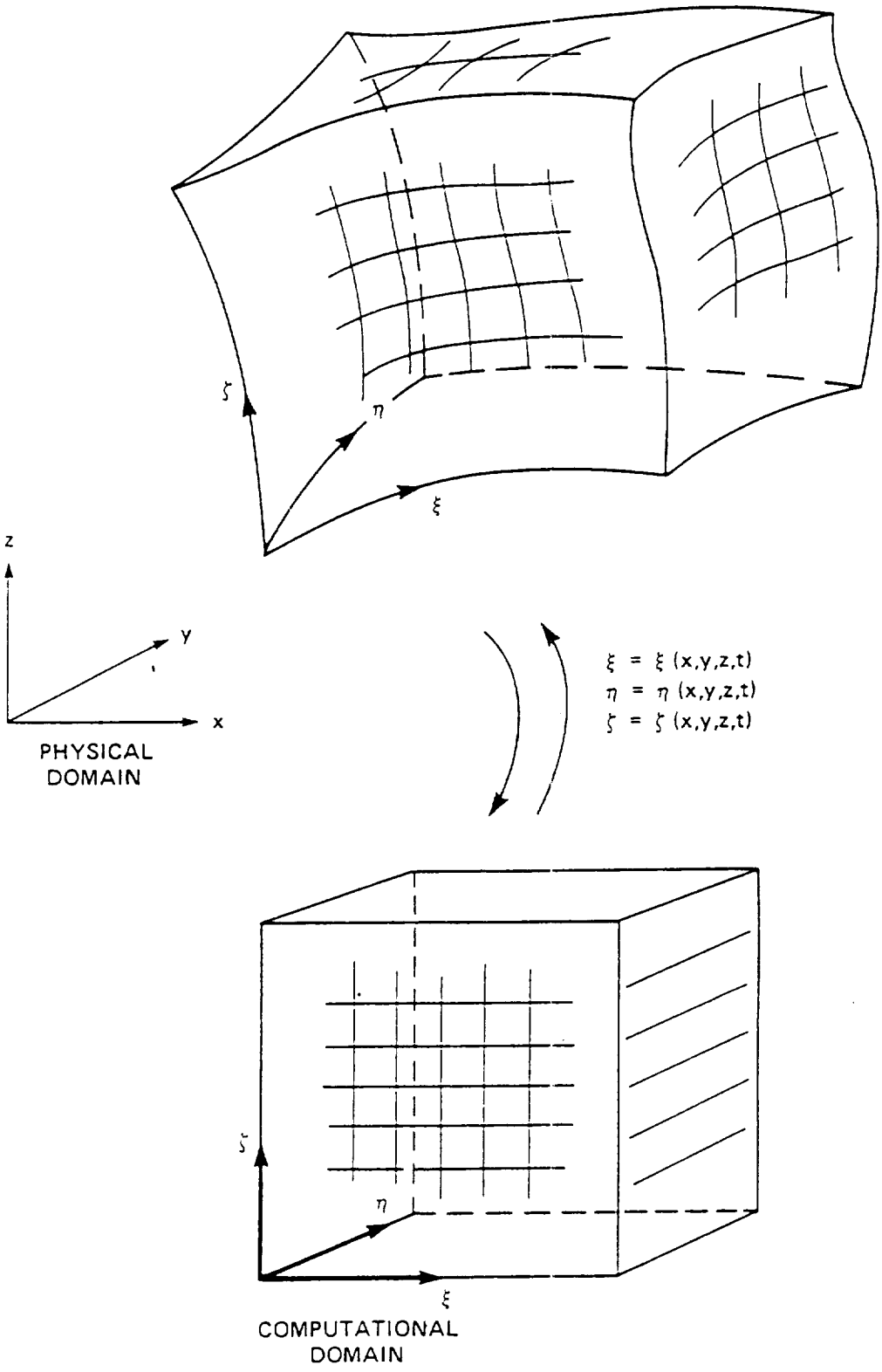


Figure 2.1. General coordinate transformation from physical to computational space (ref. 10).

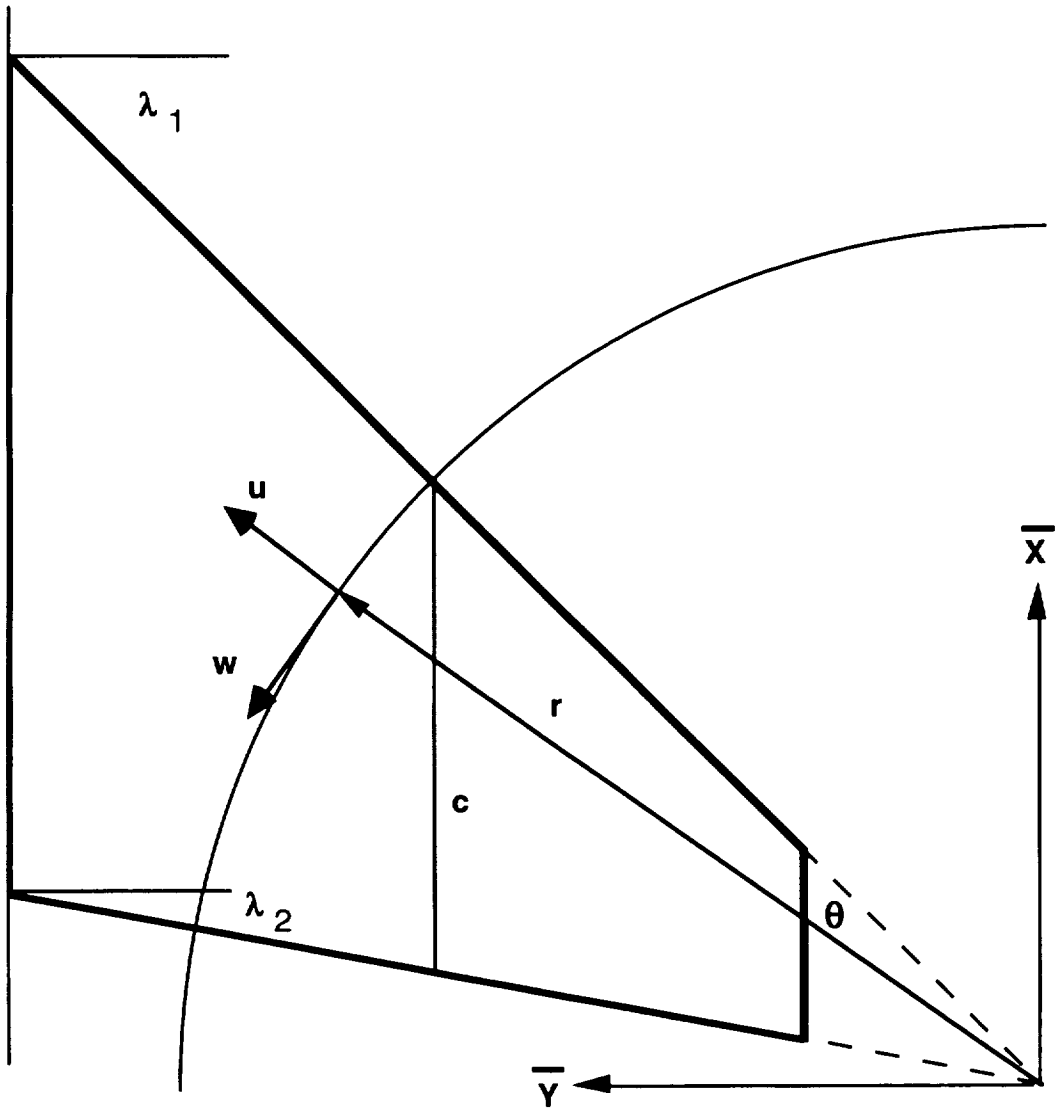
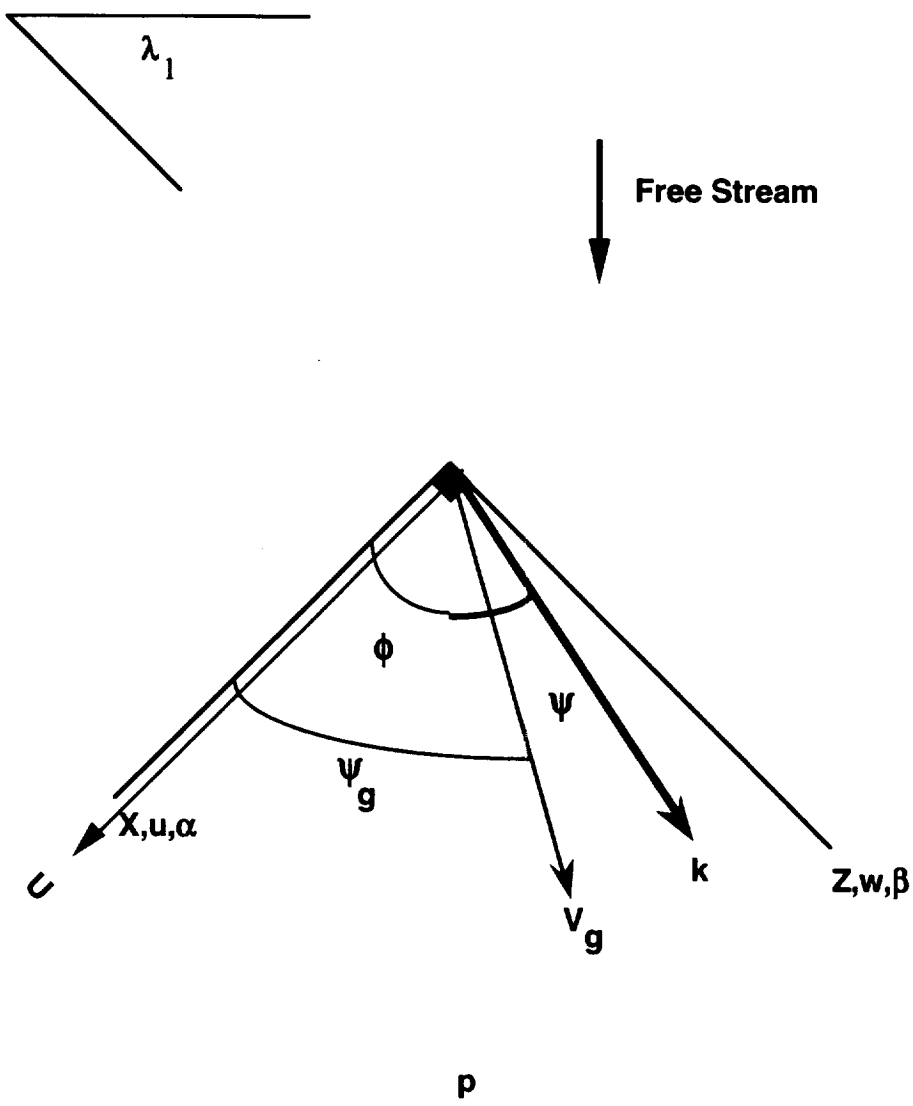


Figure 2.2. Boundary-layer code's coordinate system.



k = Disturbance Wave Vector.

a = Magnitude of the Disturbance in the **X** direction.

b = Magnitude of the Disturbance in the **Y** direction.

$\mathbf{U_p}$= Potential flow velocity.

$\mathbf{V_g}$= Group velocity (direction and speed of Disturbance wave energy).

Y = Phase Angle **k** makes w/r to the potential flow $\mathbf{U_p}$.

$\mathbf{Y_g}$ = Angle $\mathbf{V_g}$ make w/r to the **X** direction.

Figure 2.3. Disturbance wave orientation on the swept coordinate system.

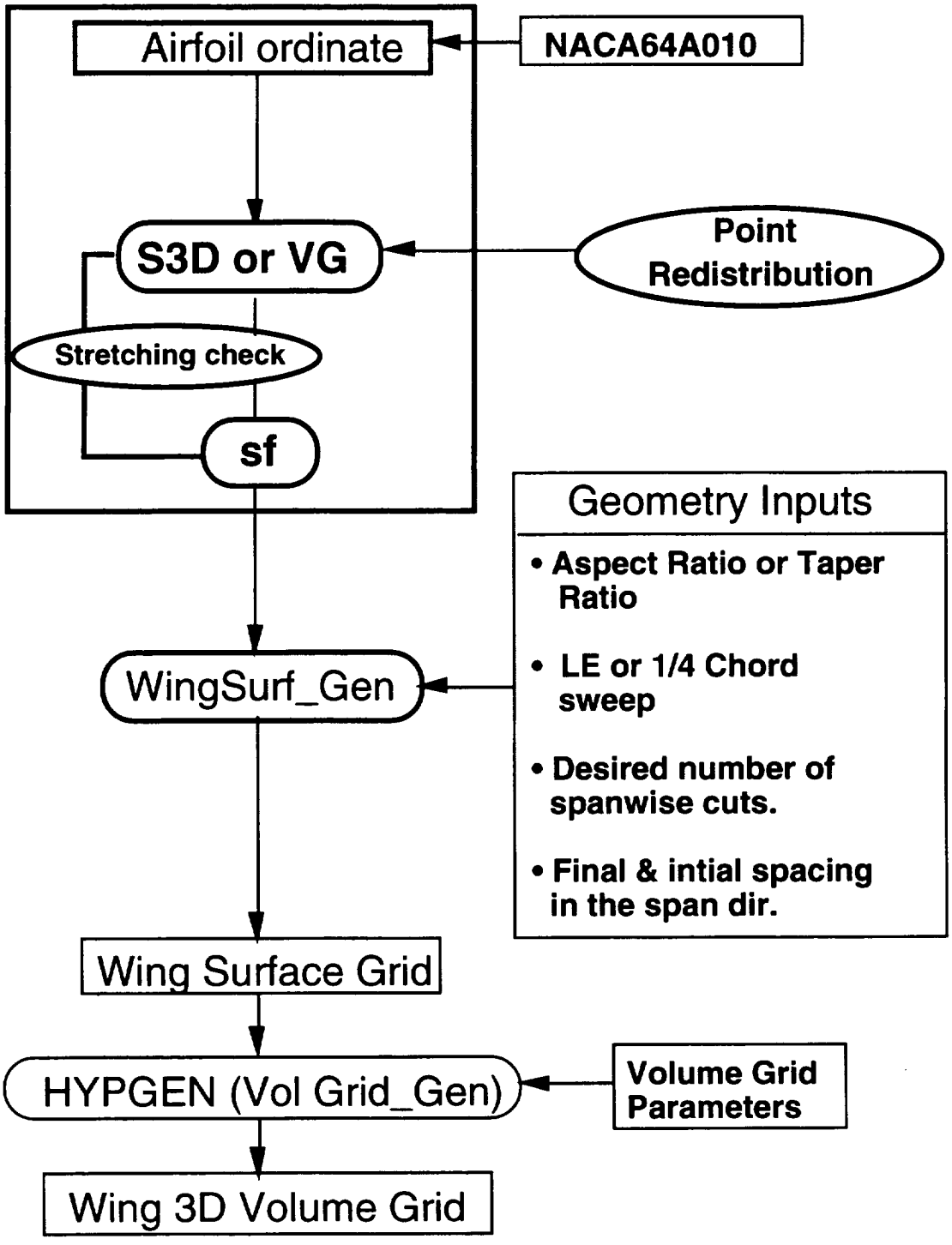


Figure 4.1. Grid generation process.

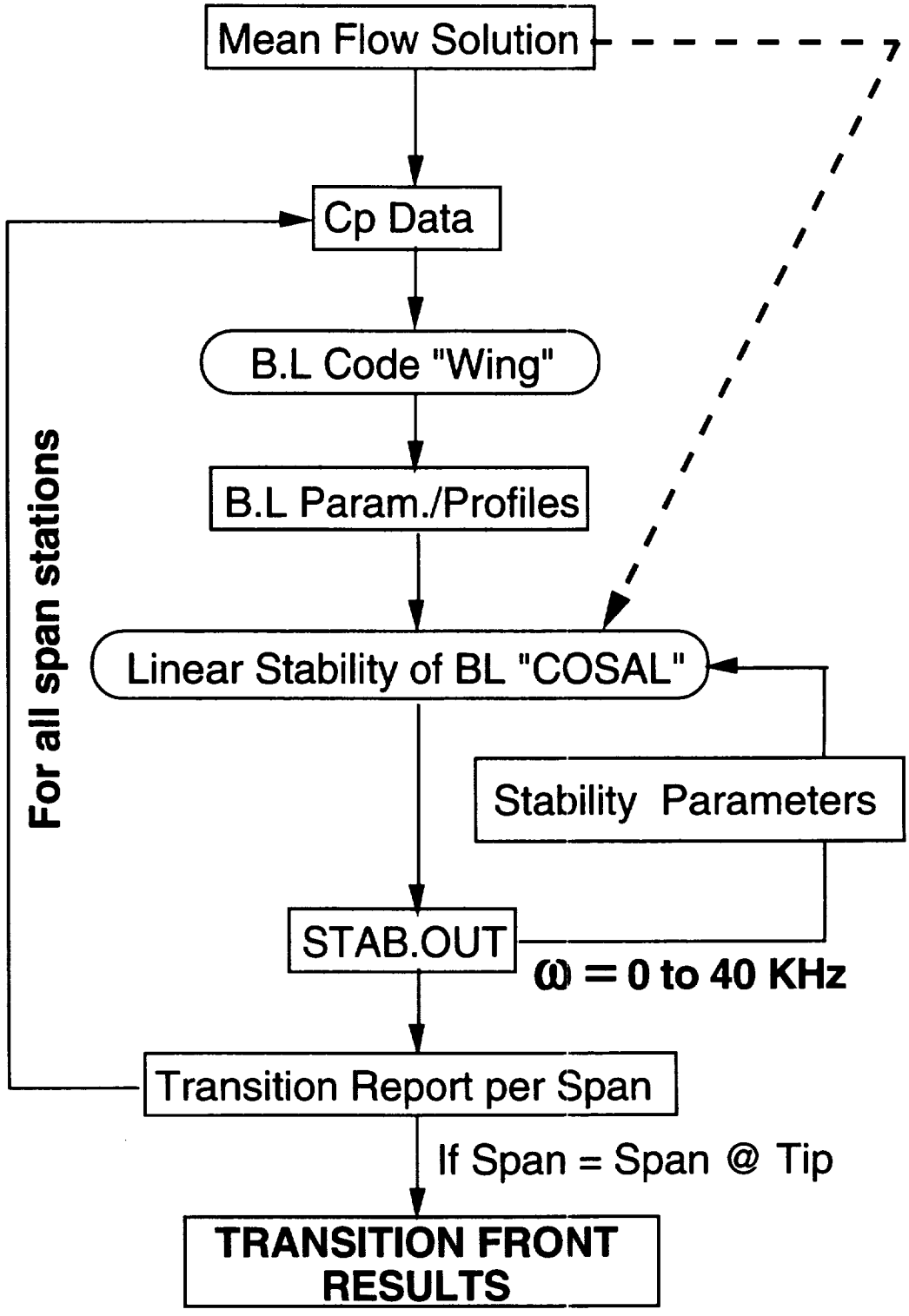


Figure 5.1. Automated stability analysis process.

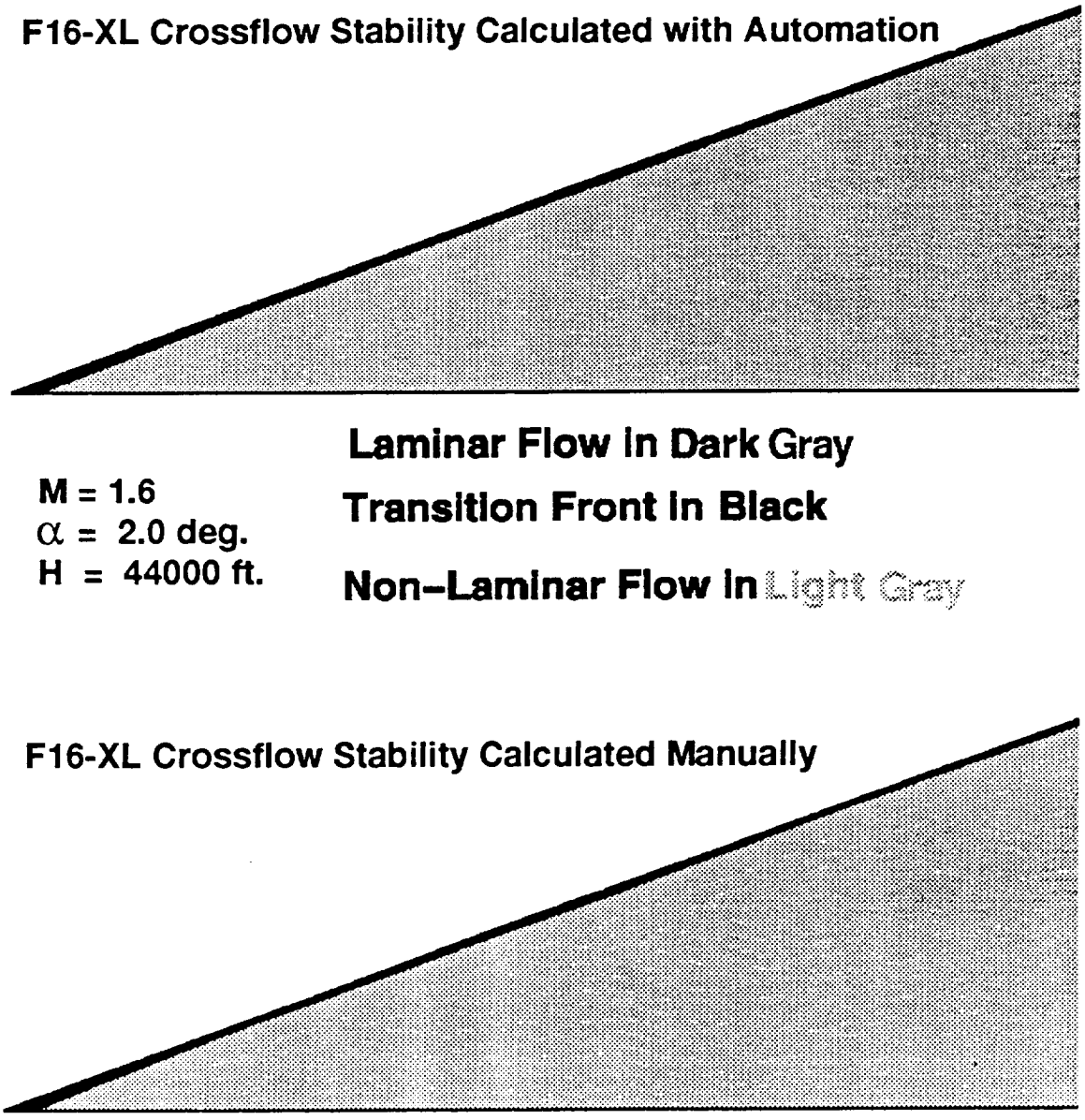


Figure 6.1. Stability automation validation.

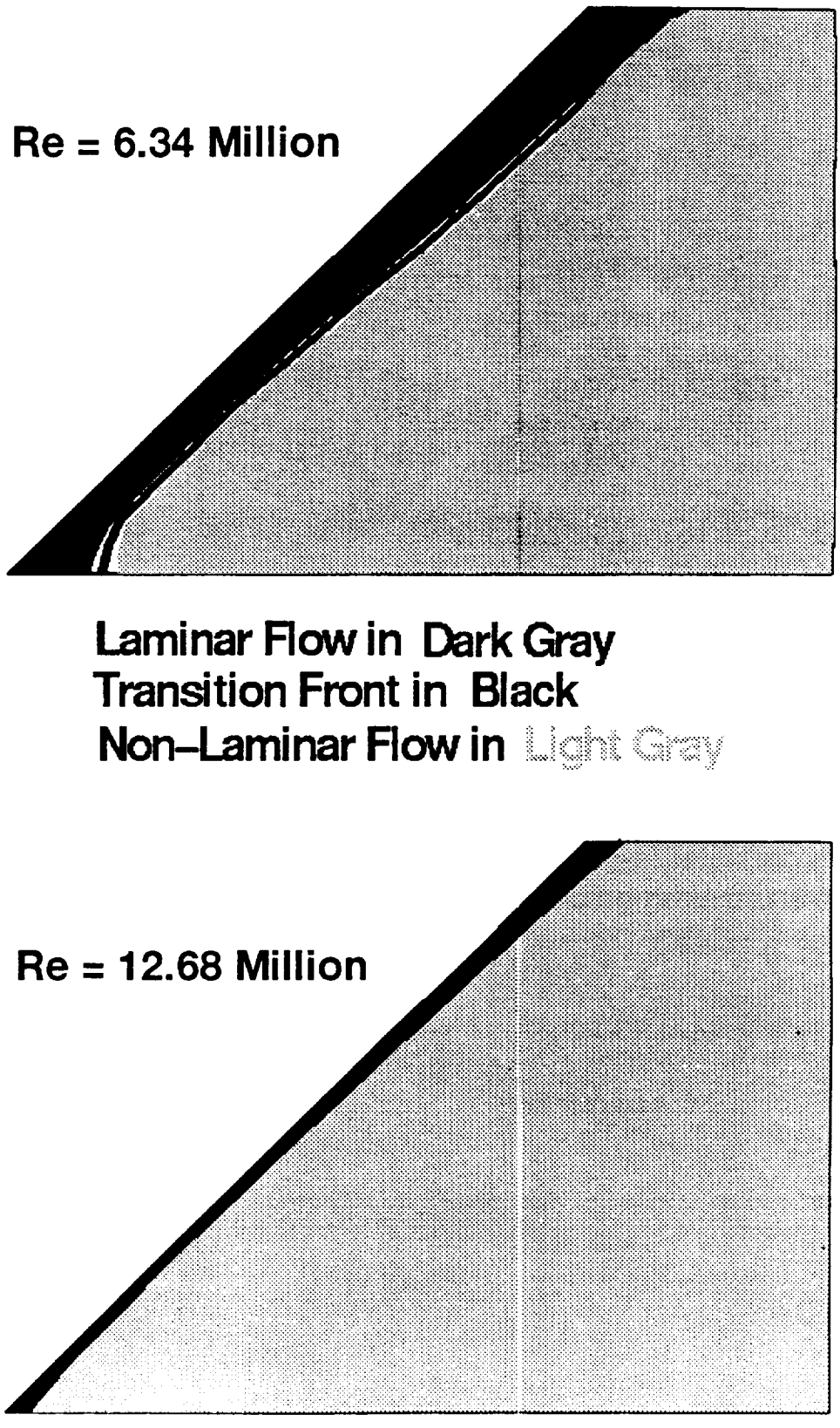


Figure 6.2. Transition front result due to Reynolds number.

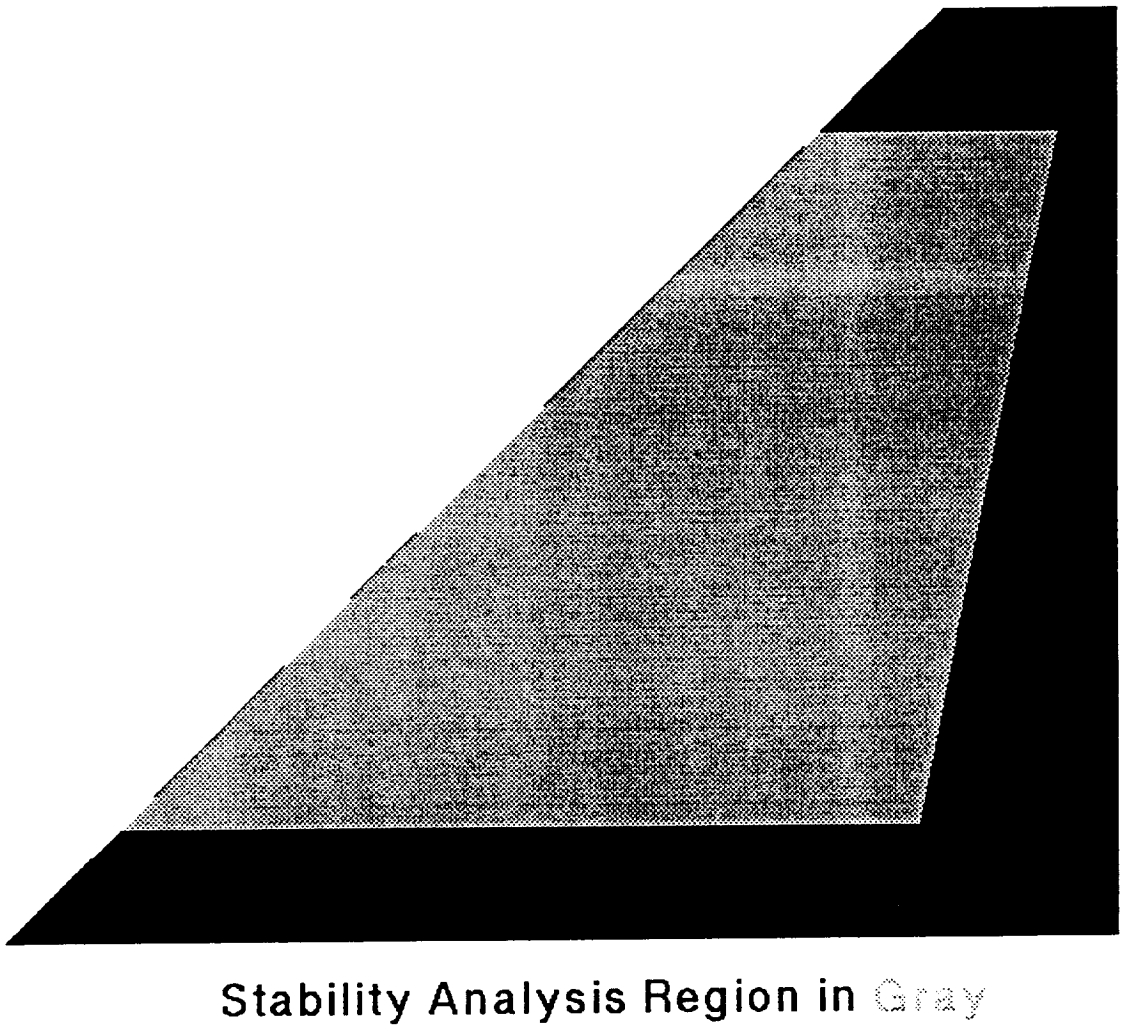


Figure 6.3. Boundary-layer stability analysis region.

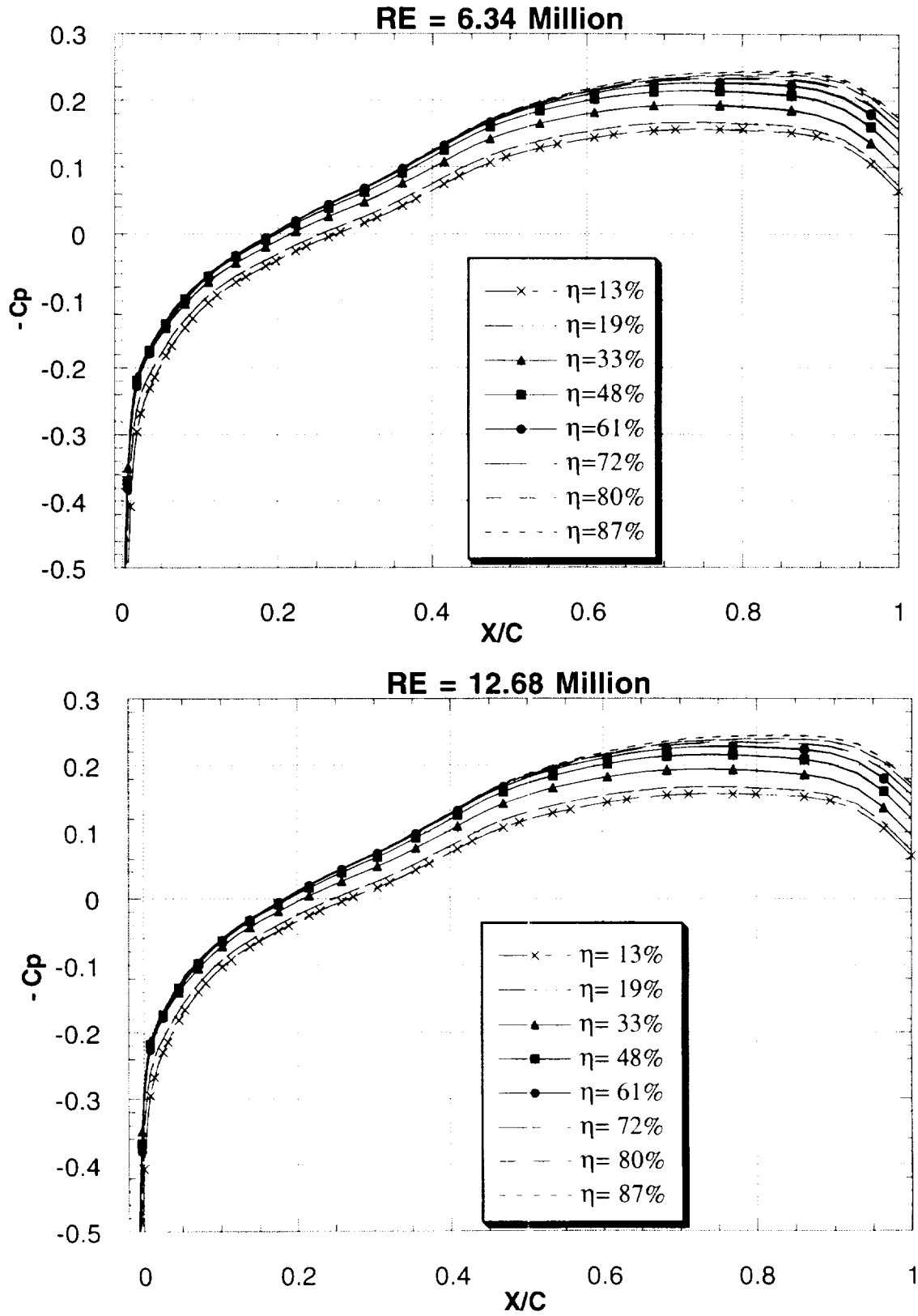


Figure 6.4. Chordwise pressure distribution (a = 0).

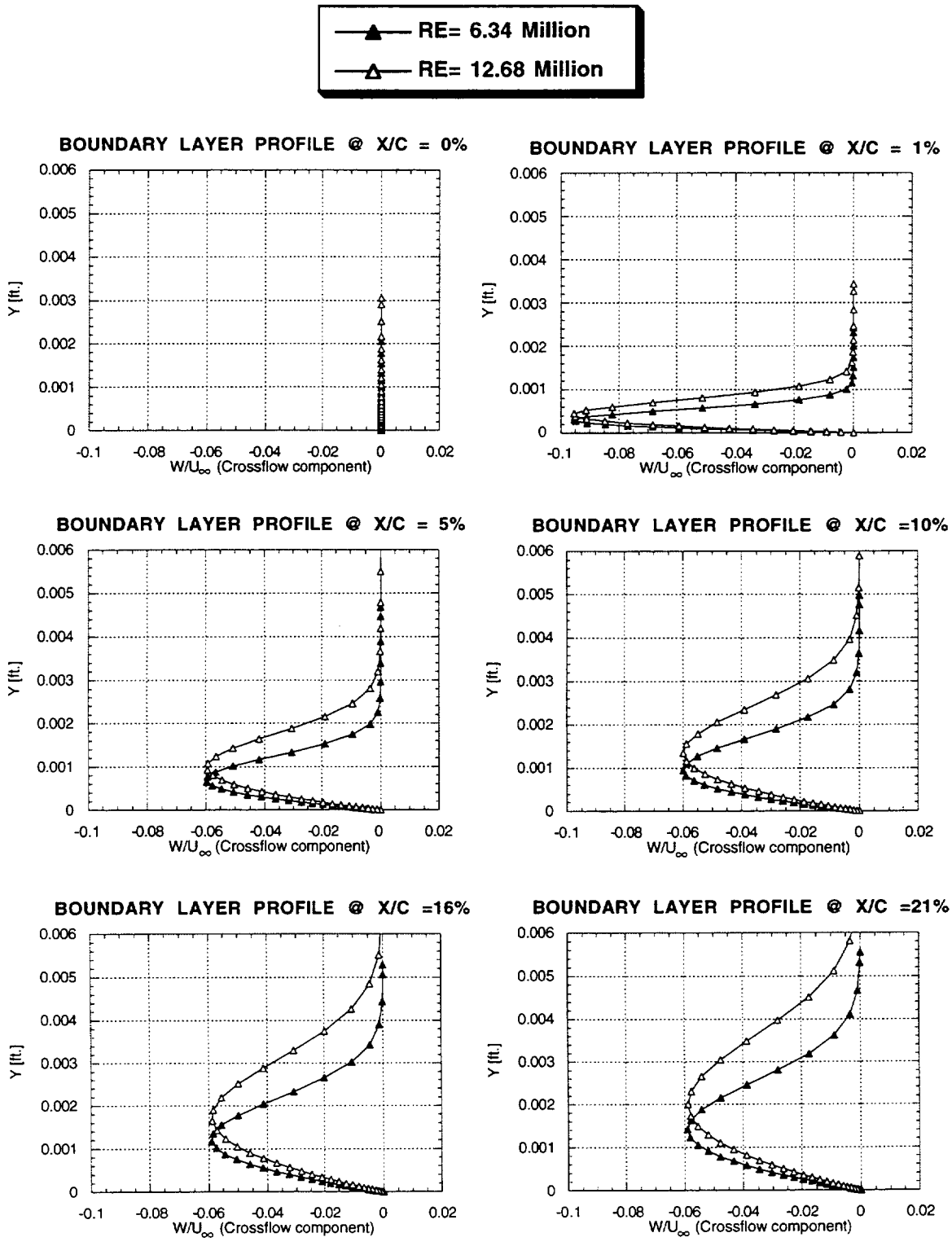


Figure 6.5. Effect of Reynolds number on crossflow at 48% semispan.

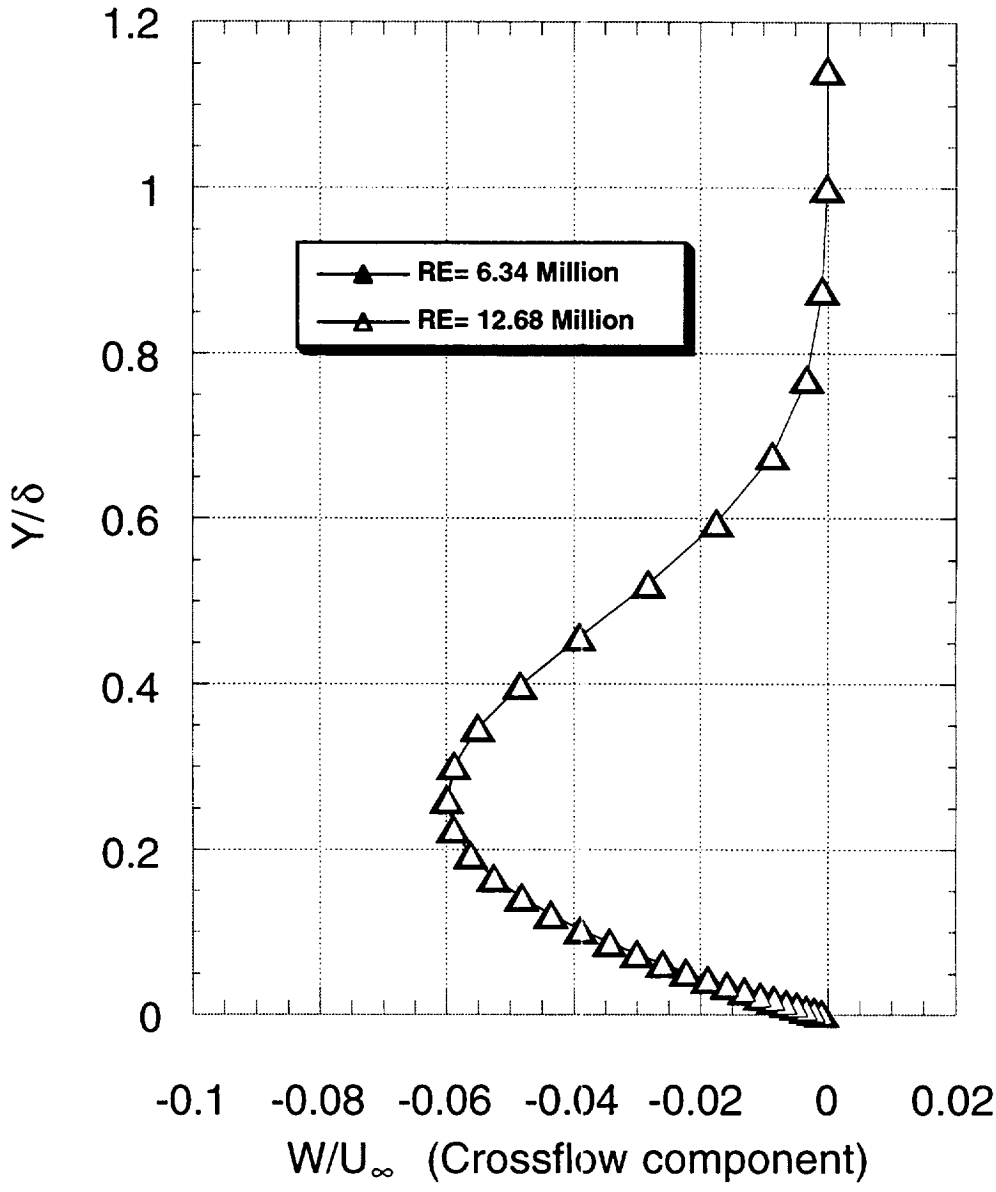


Figure 6.6. Effect of Reynolds number on crossflow at 48% semispan for x/c = 10%.

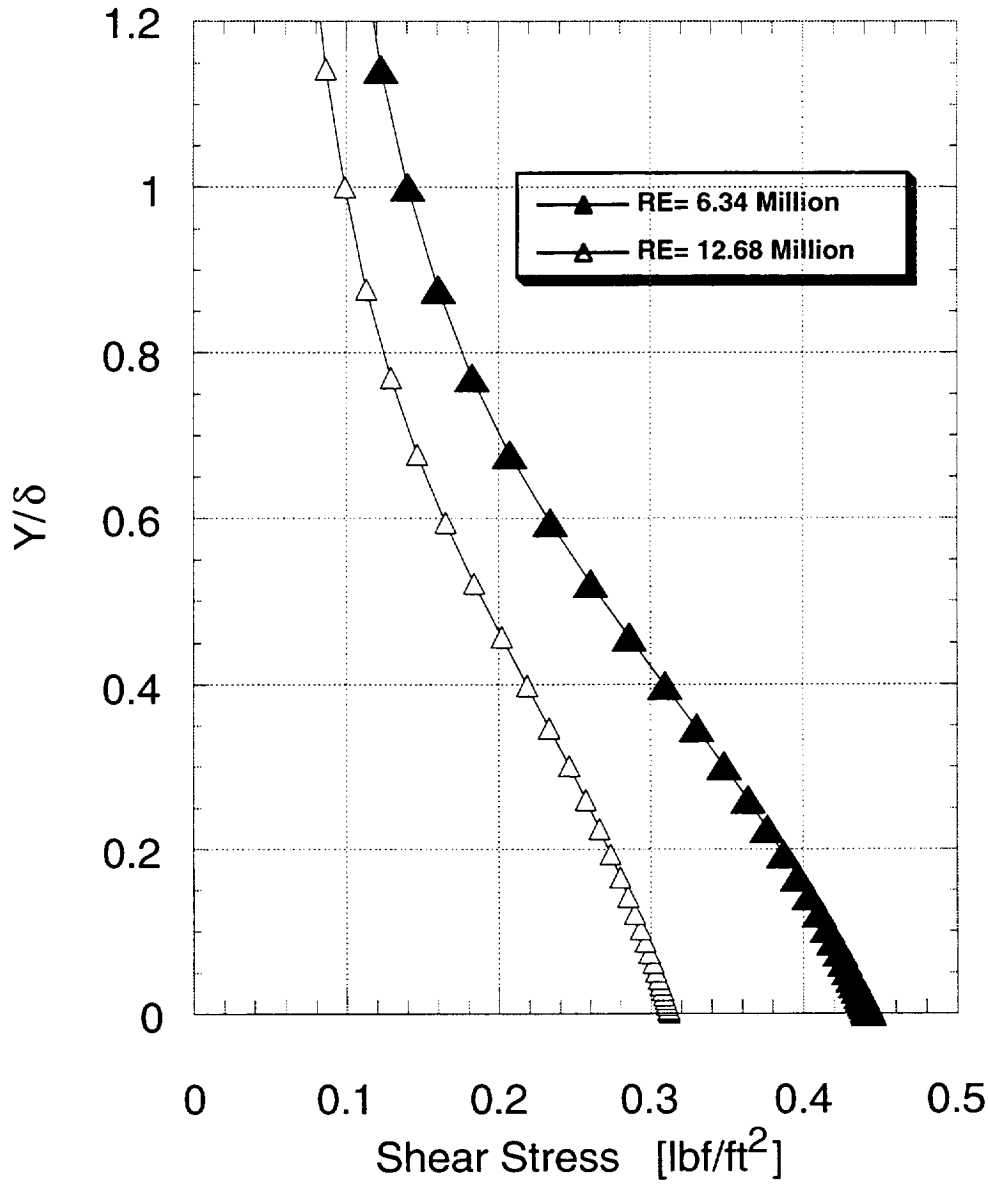


Figure 6.7. Effect of Reynolds number on shear stress in the boundary-layer at 48% semispan for x/c = 10%.

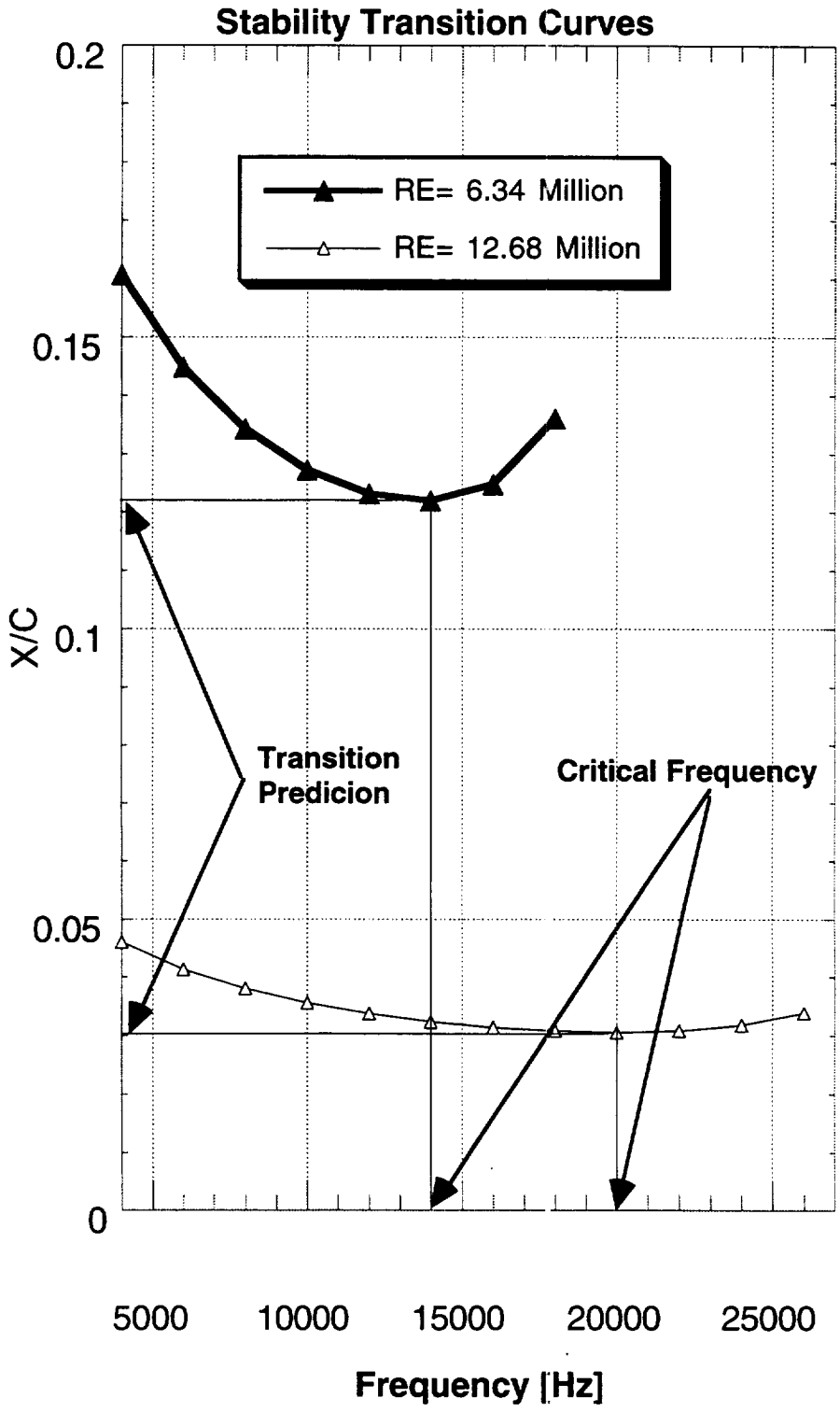


Figure 6.8. Effect of Reynolds number on transition at 48% semispan.

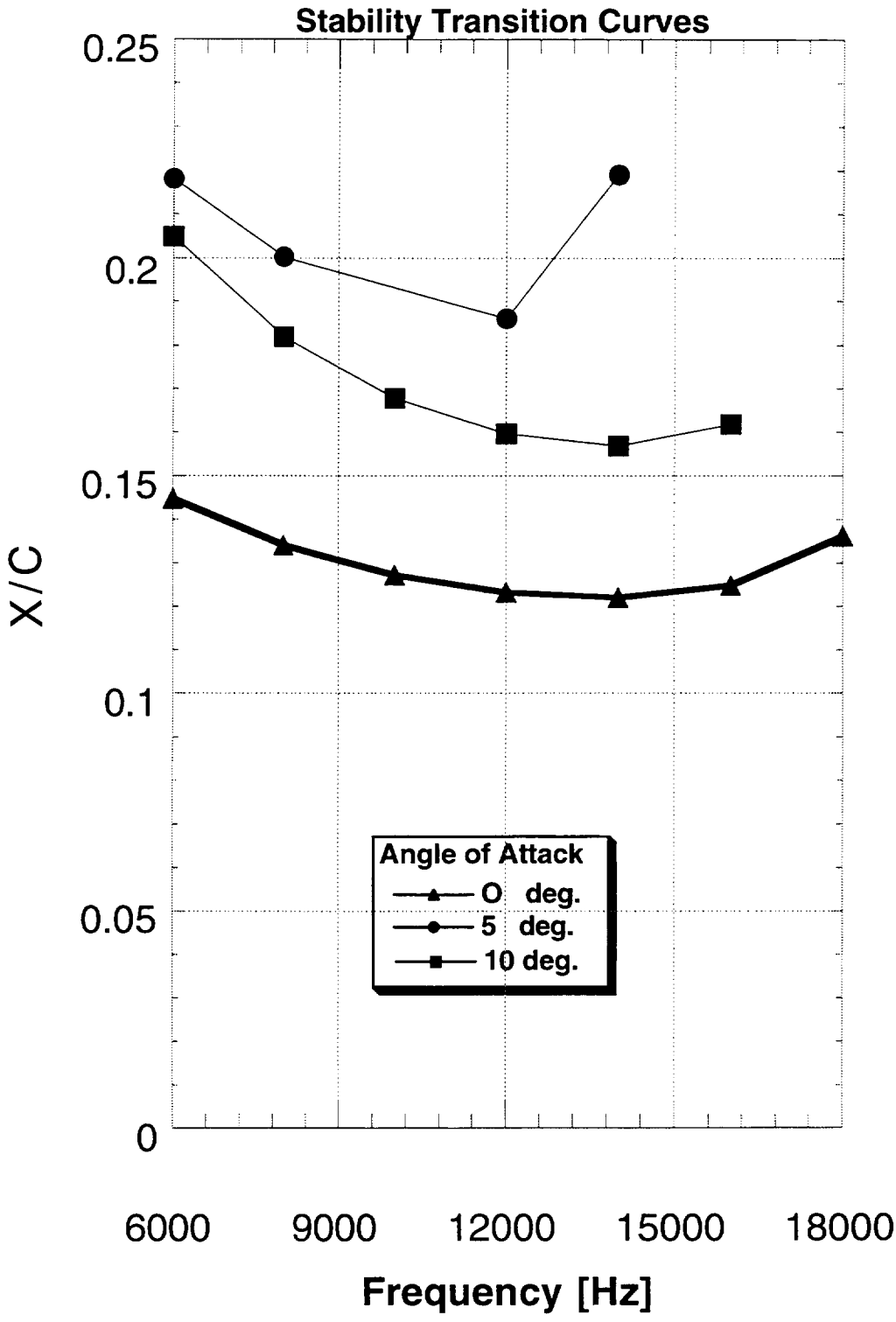


Figure 6.9. Effect of angle of attack on transition prediction at 48% semispan for Re = 6.34 million and 45° sweep.

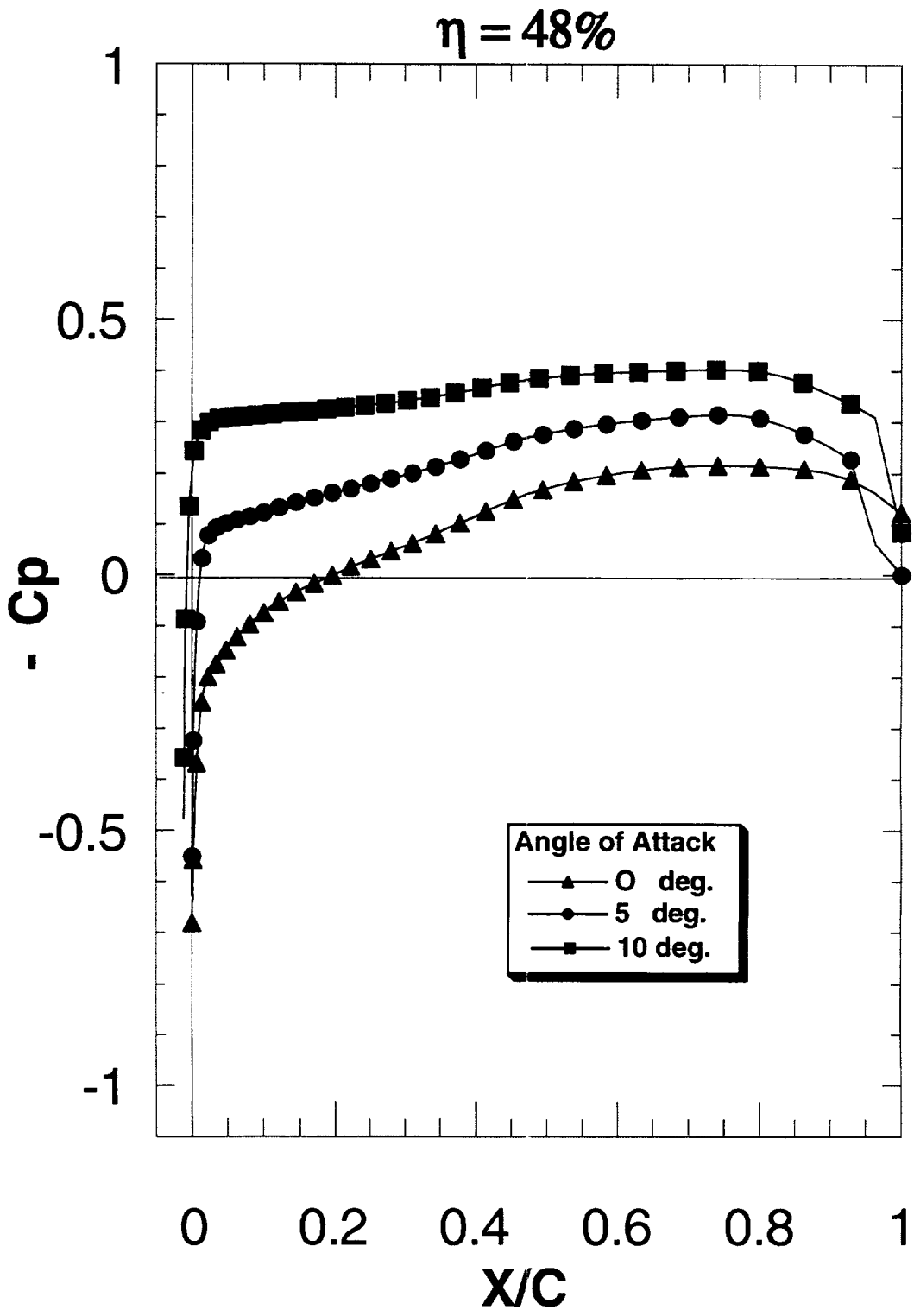


Figure 6.10. Chordwise pressure distribution effects at 48% semispan due to angle of attack at Re = 6.34 million.

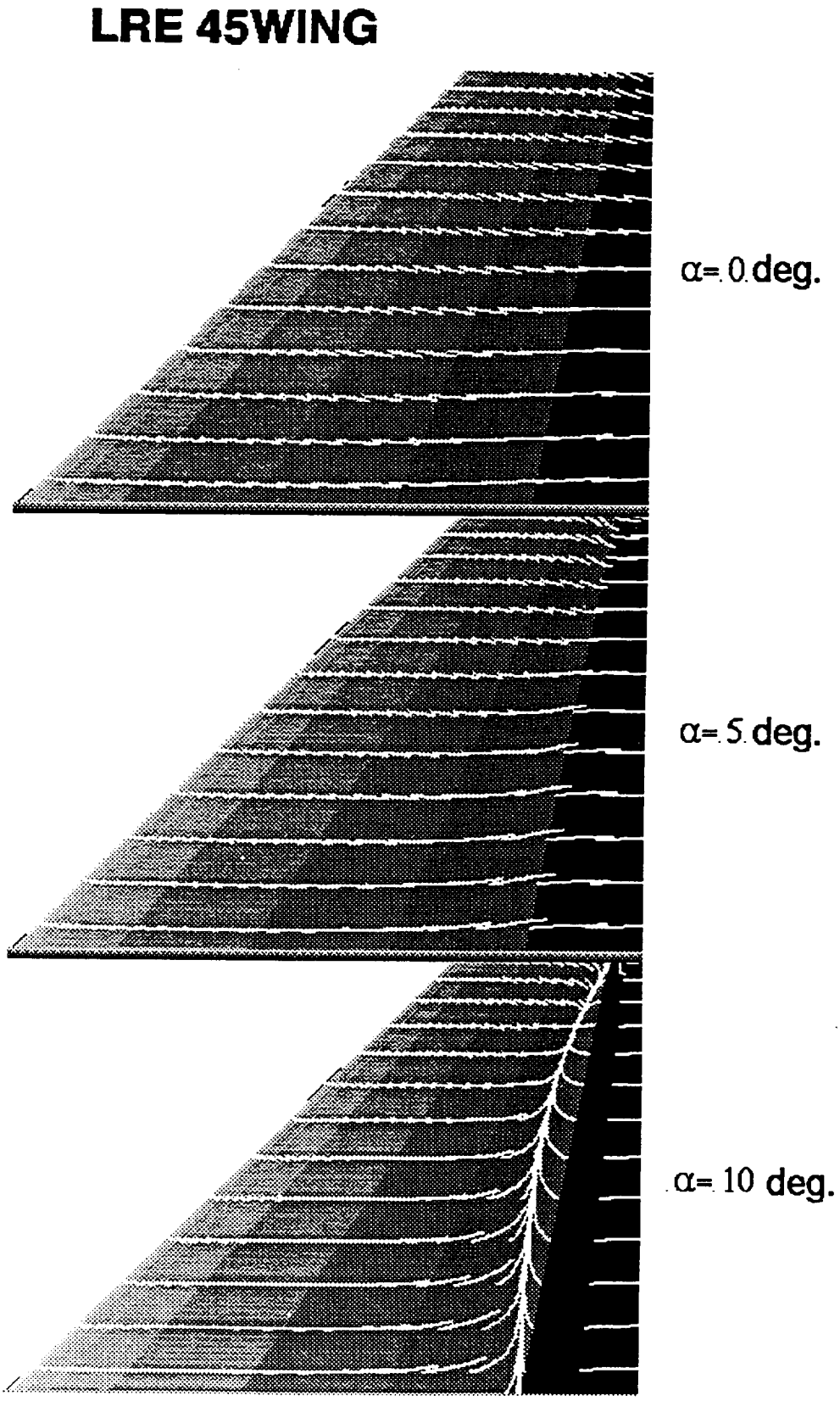


Figure 6.11. Effect of angle of attack on surface flow patterns.

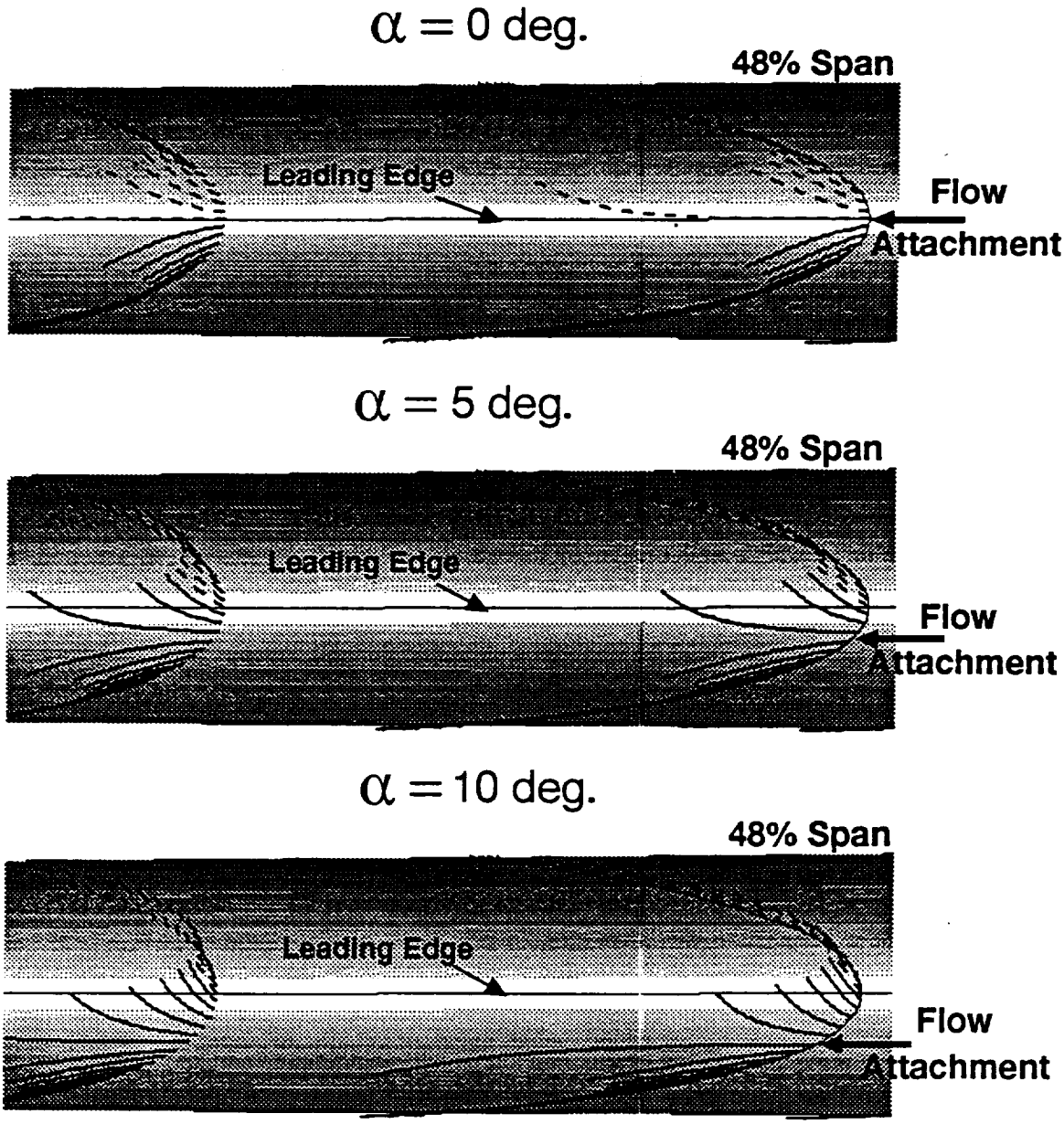


Figure 6.12. Effect of angle of attack on leading edge flow attachment at 48% semispan.

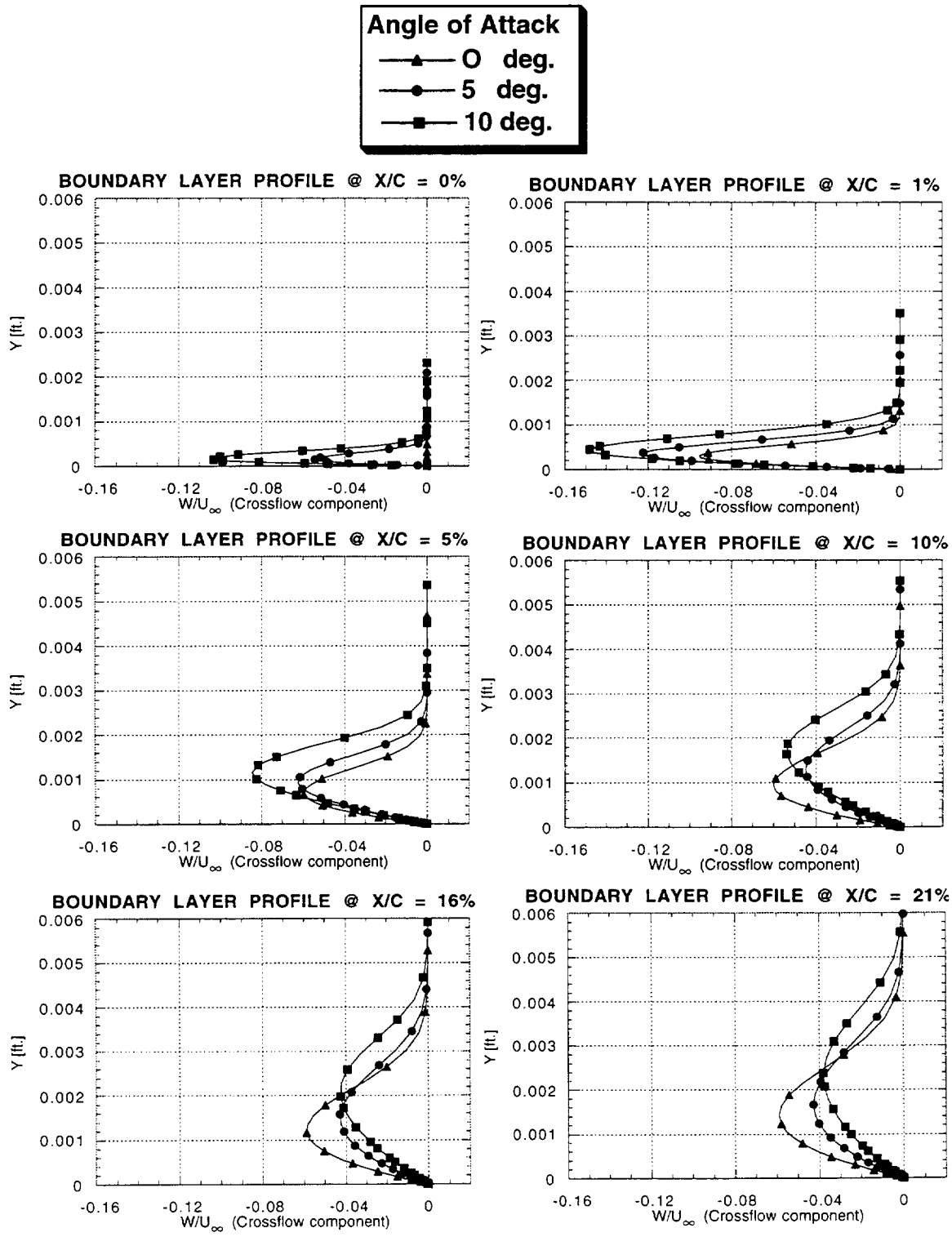


Figure 6.13. Effect of angle of attack on crossflow profiles at 48% semispan for Re = 6.34 million and 45° sweep.

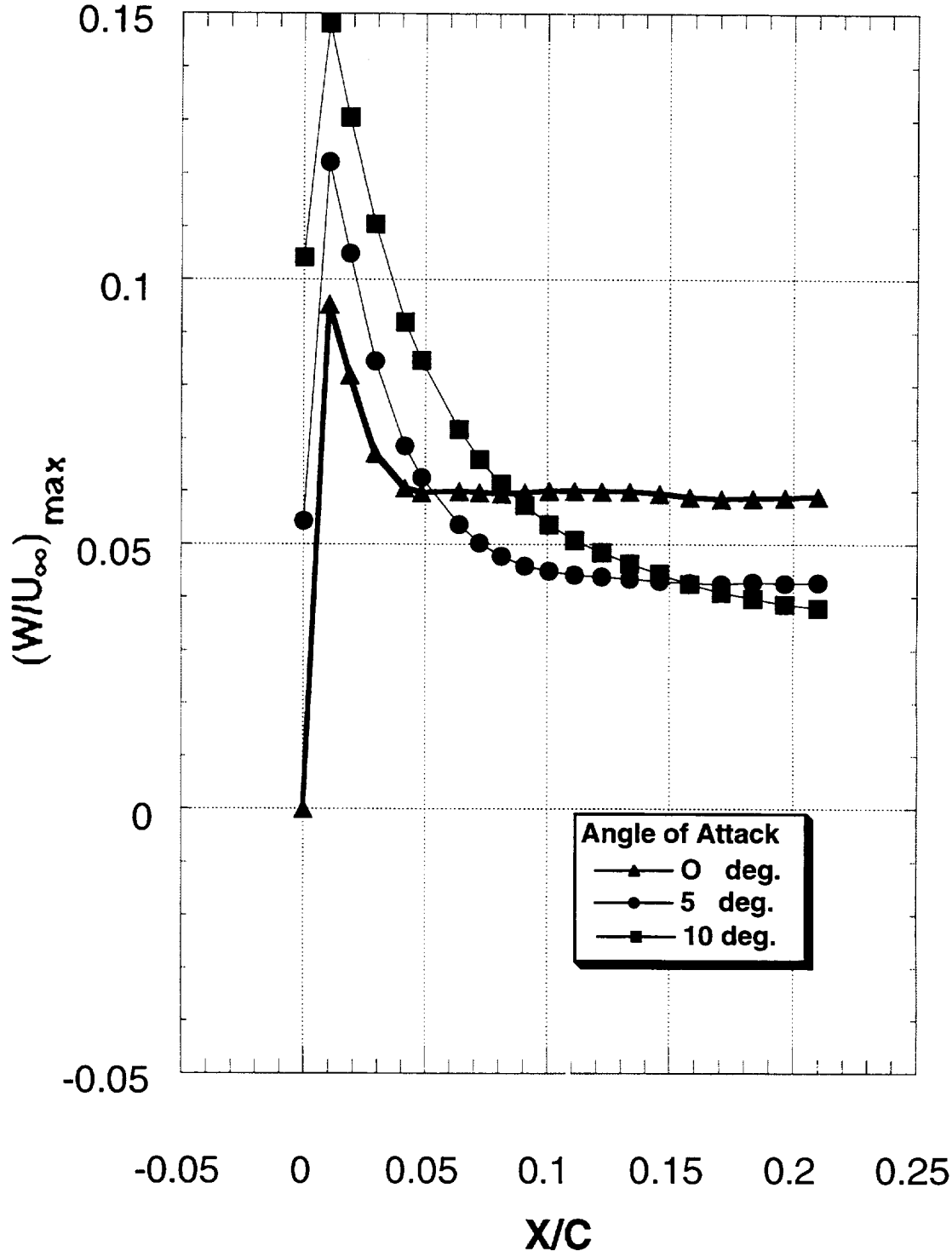


Figure 6.14. Maximum crossflow effect due to angle of attack at 48% semispan for Re = 6.34 million and 45° sweep.

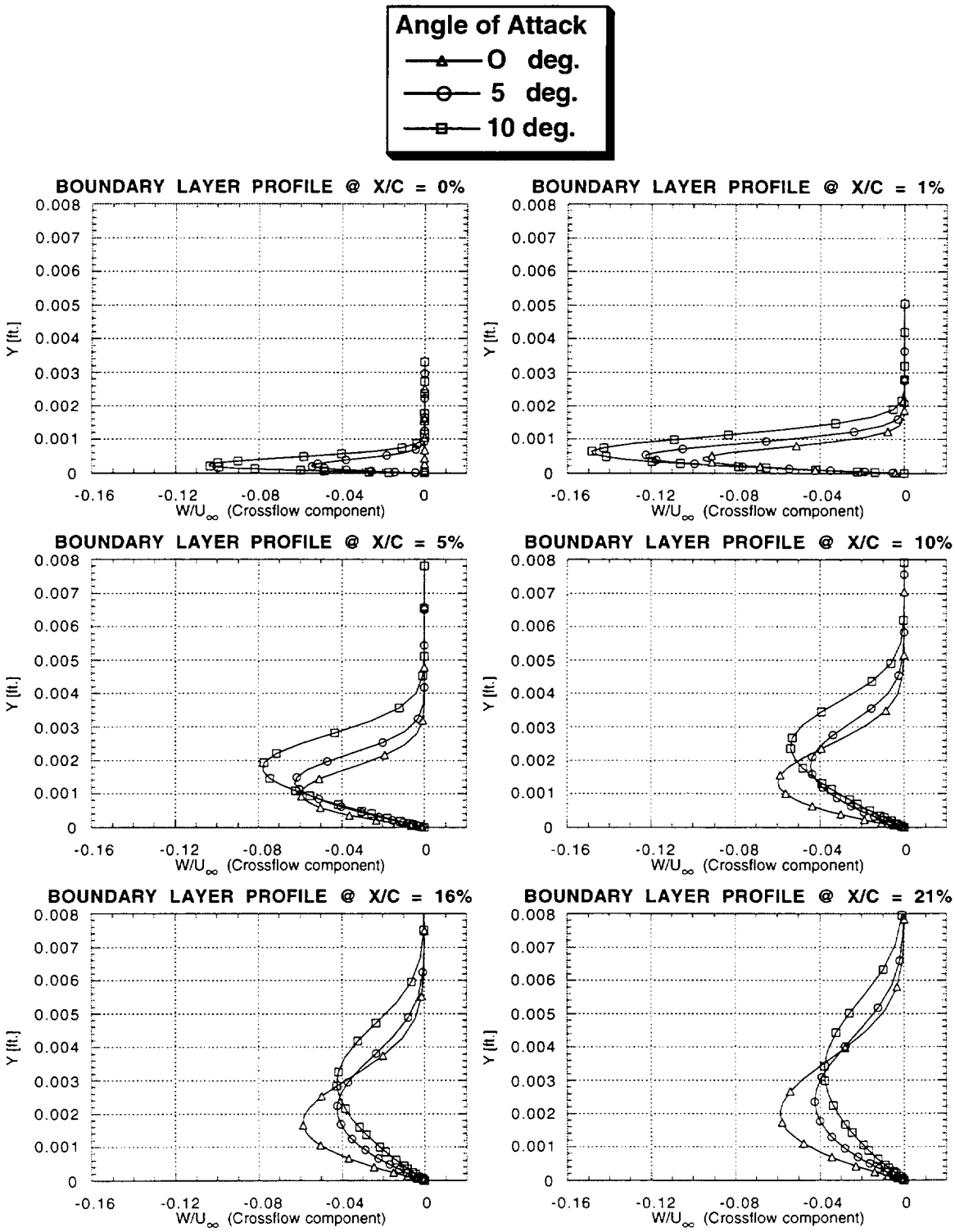


Figure 6.15. Effect of angle of attack on crossflow at 48% semispan for Re = 12.68 million and 45° sweep.

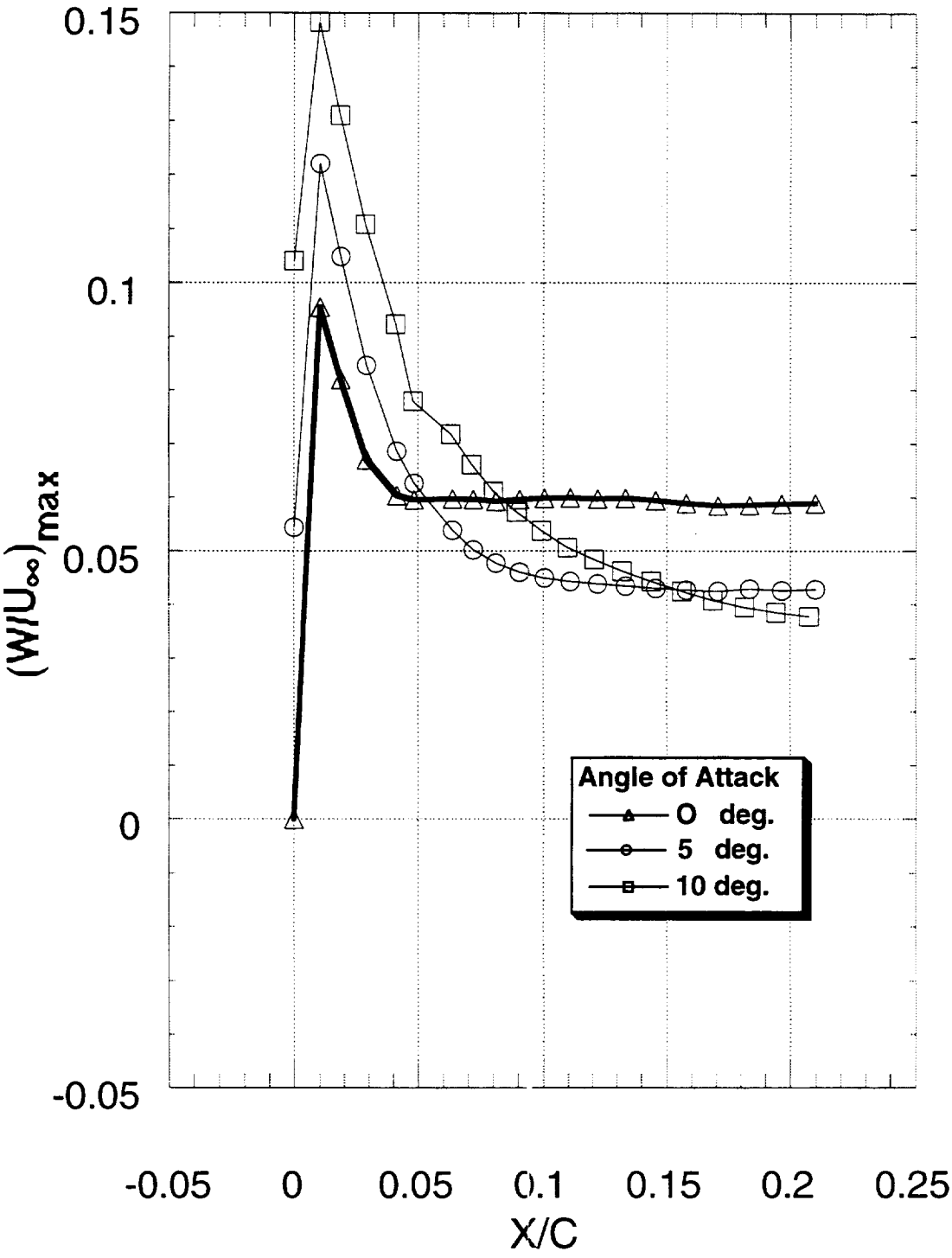


Figure 6.16. Maximum crossflow effect due to angle of attack at 48% semispan for Re = 12.68 million and 45° sweep.

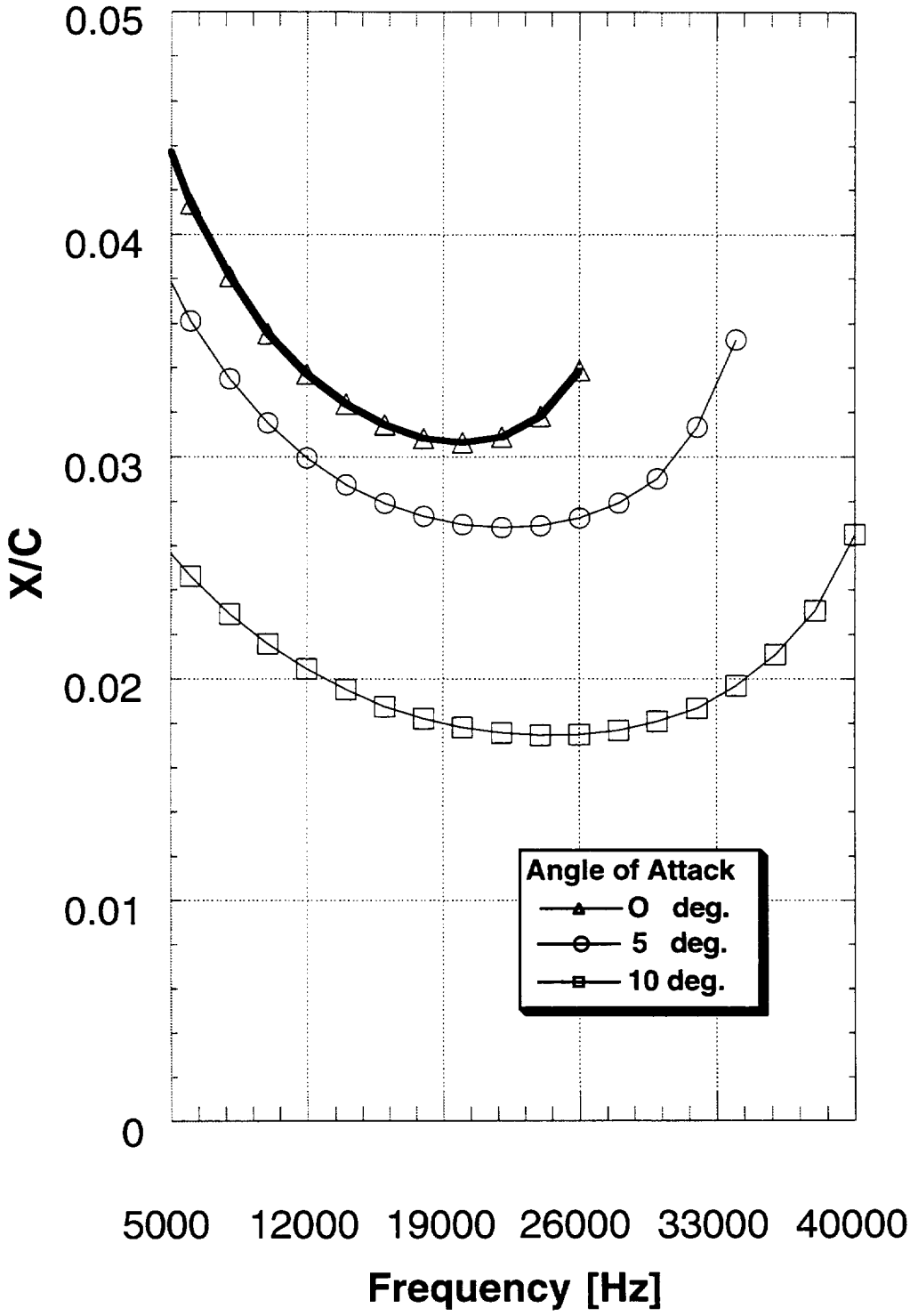
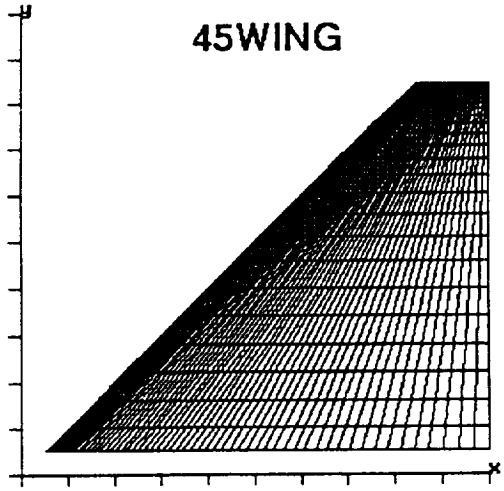


Figure 6.17. Higher Reynolds number effect with angle of attack on transition for the 45° sweep.

Leading Edge Sweep = 45 deg.

Trailing Edge Sweep = 0 deg.

Aspect Ratio = 1.45



Leading Edge Sweep = 60 deg.

Trailing Edge Sweep = 36.2 deg.

Aspect Ratio = 1.45

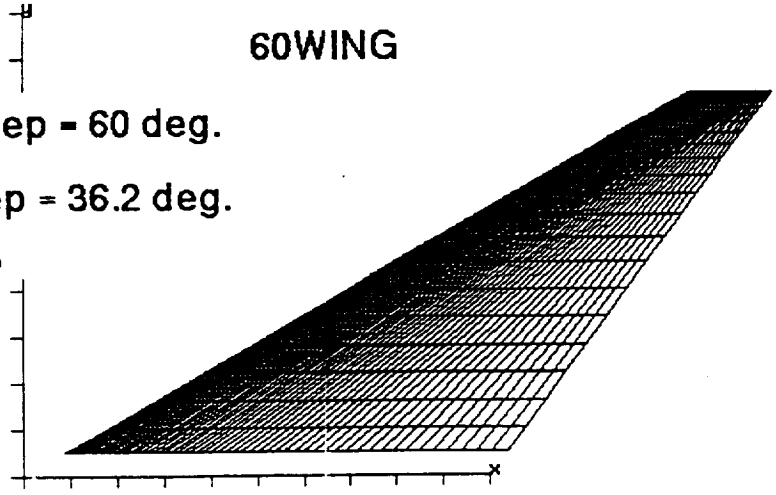


Figure 6.18. Swept geometry surface grids.

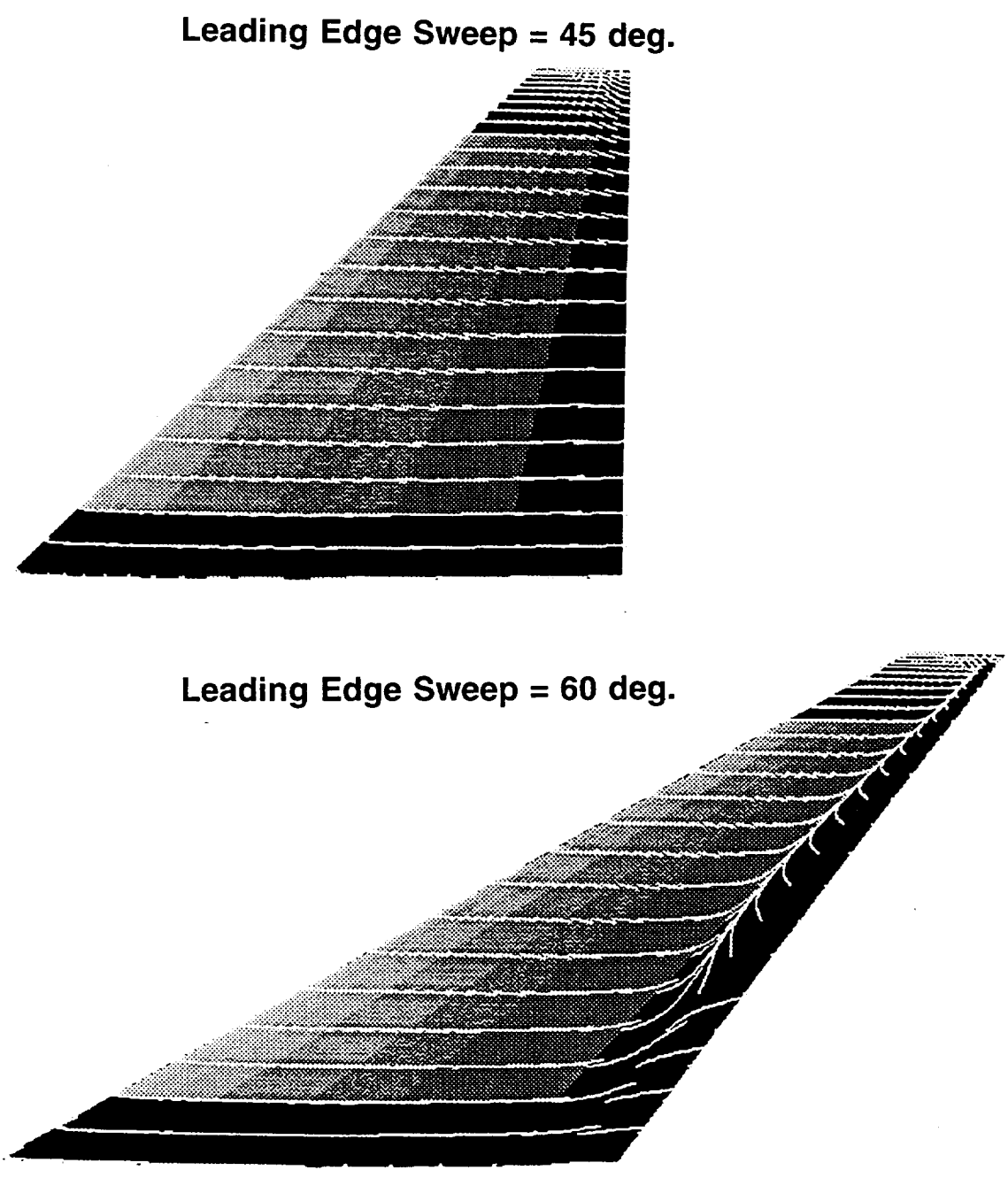


Figure 6.19. Effect of sweep on surface flow patterns at the lower Reynolds number and 0° angle of attack case.

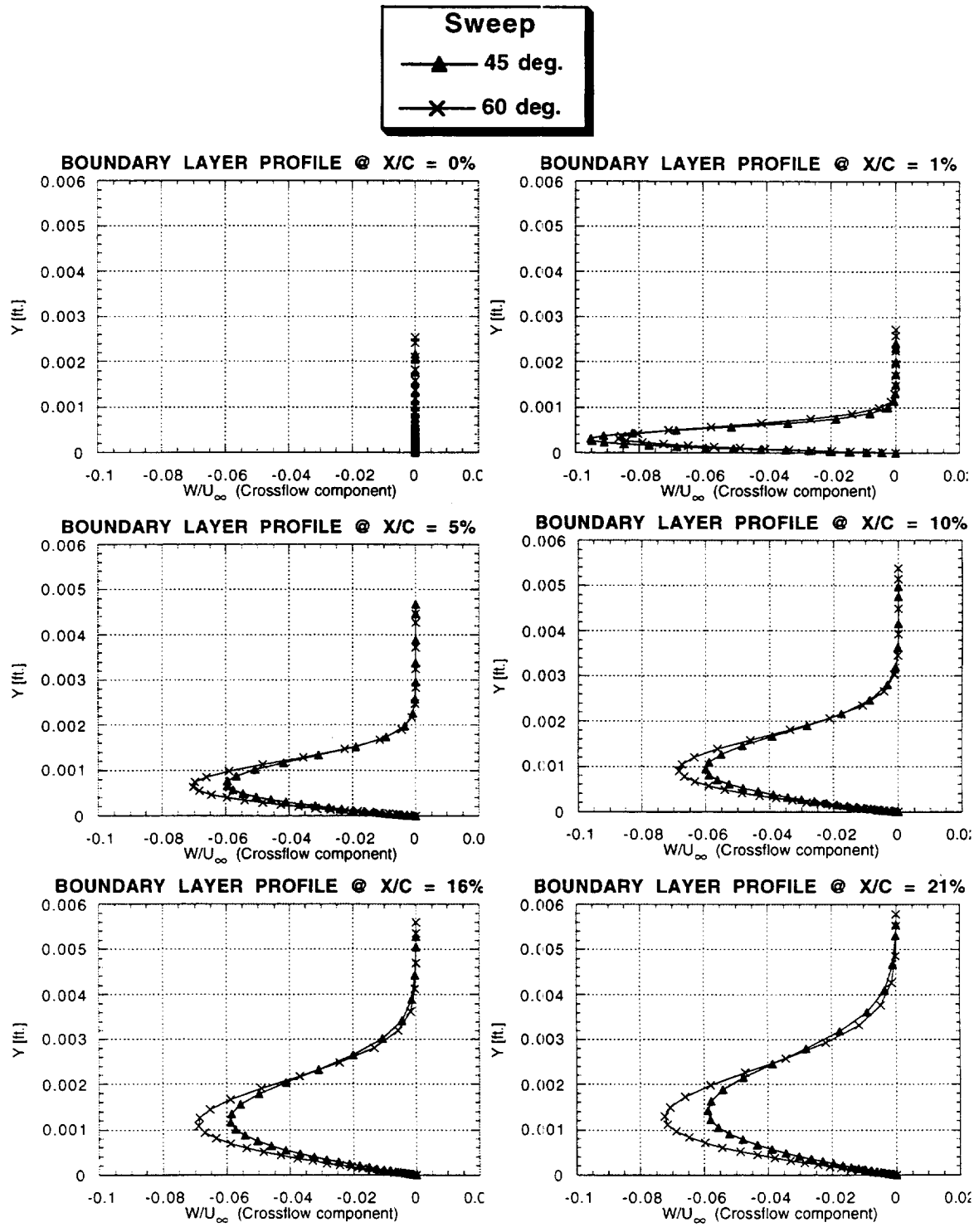


Figure 6.20. Effect of leading edge sweep on crossflow profiles at 48% semispan (Re = 6.34 million and a = 0°).

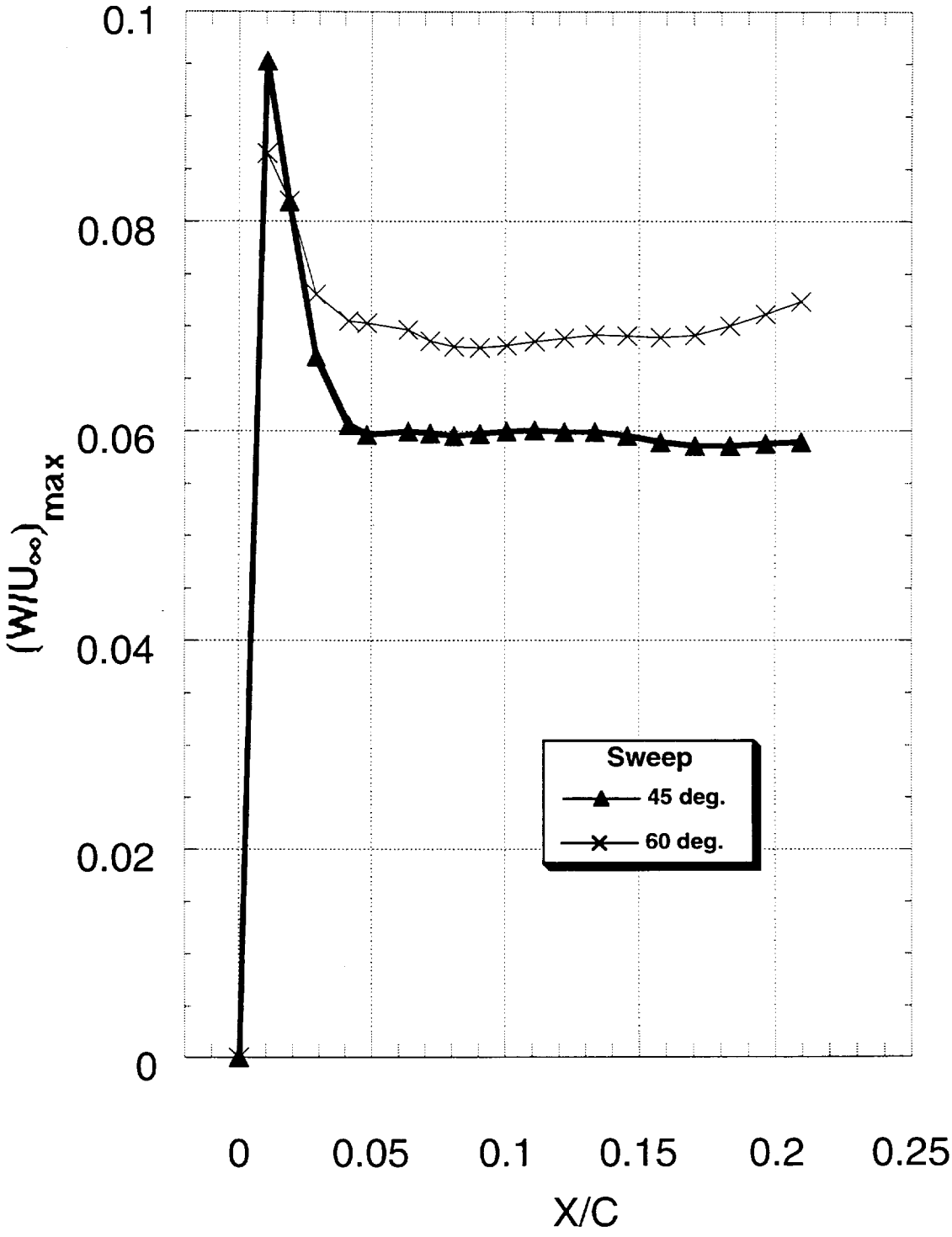


Figure 6.21. Maximum crossflow effect due to leading edge sweep at 48% semispan for Re = 12.68 million and $\alpha = 0^{\circ}$.

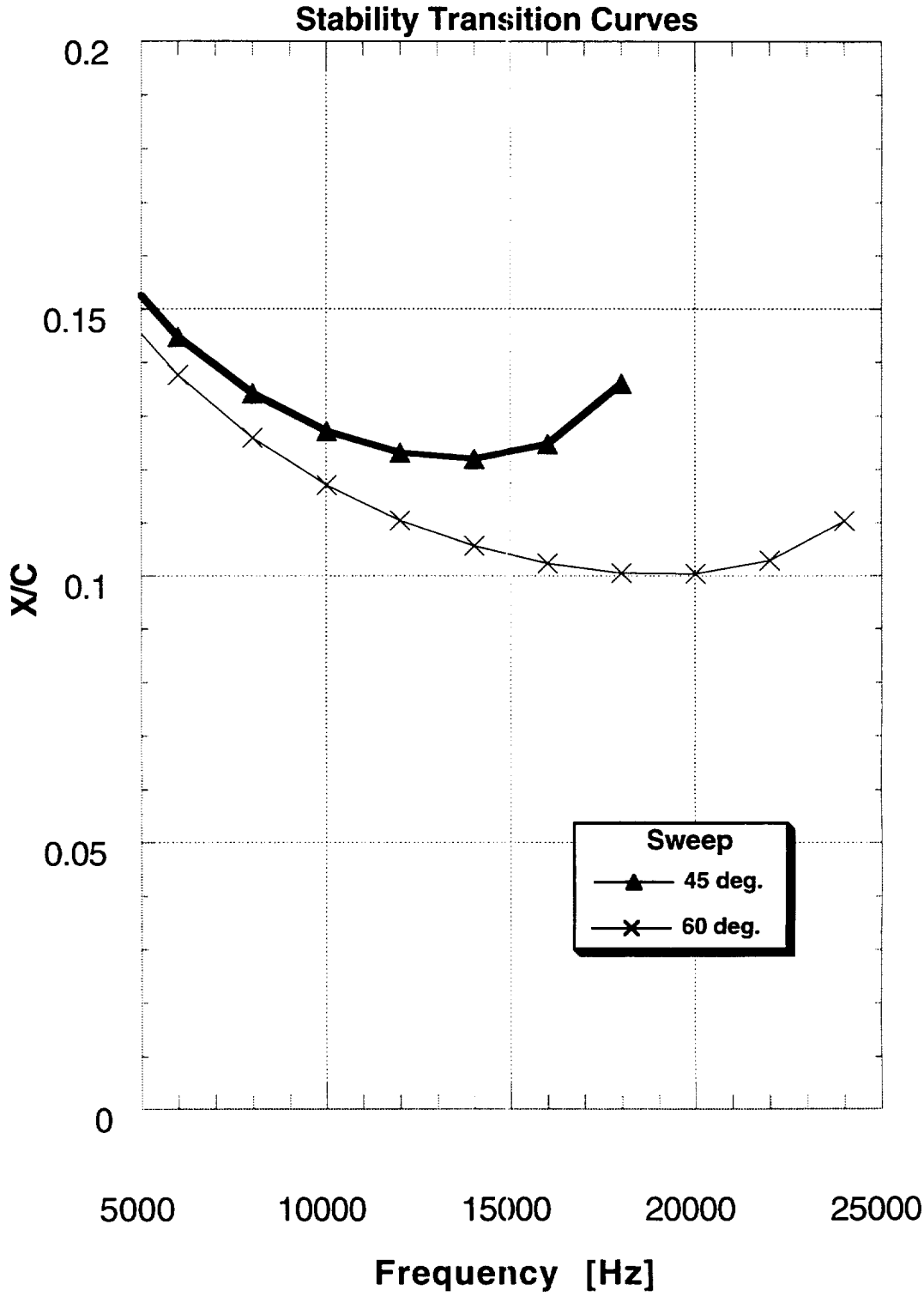


Figure 6.22. Effect of leading edge sweep on transition at 48% semispan for Re = 12.68 million and $\alpha = 0^{\circ}$.

APPENDIX A

WING SURFACE GRID CREATION PROCEDURE

The following will describe the process used to generate a surface grid for any NACA 6- or 6a-series airfoil.

Steps

I. Run the 6-series code "sixseries.f" (ref. 18) with the proper input file to get an output file called "fort.10" containing the airfoil ordinates.

NOTE: THE FOLLOWING STEPS WILL DEPEND ON WHETHER YOU ARE GOING TO USE VG OR S3D TO REDISTRIBUTE THE POINTS:

II. FOR VG:

1) Use the program "airf_2dsurf.f", which will take the sixseries airfoil ordinates output and create a file with just the upper surface ordinates of the airfoil. The output file will be called "airf.crv".
2) Now run the code Visual Grid on the "airf.crv" file to cluster points at the L.E. and T.E. Note, every time you redistribute the point write an output file called "____.crv" and check to see that the stretching factor is less than 1.3 [sf < 1.3]. This is done by editing the ___.crv file so only the newly redistributed points are in the file and then running the program "sf.f", which read the "___.crv" file and checks each point to see if it meets the criteria of sf < 1.3. Once the point distribution meets the criteria you now have the output file "___.crv" which is the correctly distributed upper surface airfoil ordinates.

THINGS TO REMEMBER ABOUT REDISTRIBUTING ON VG:

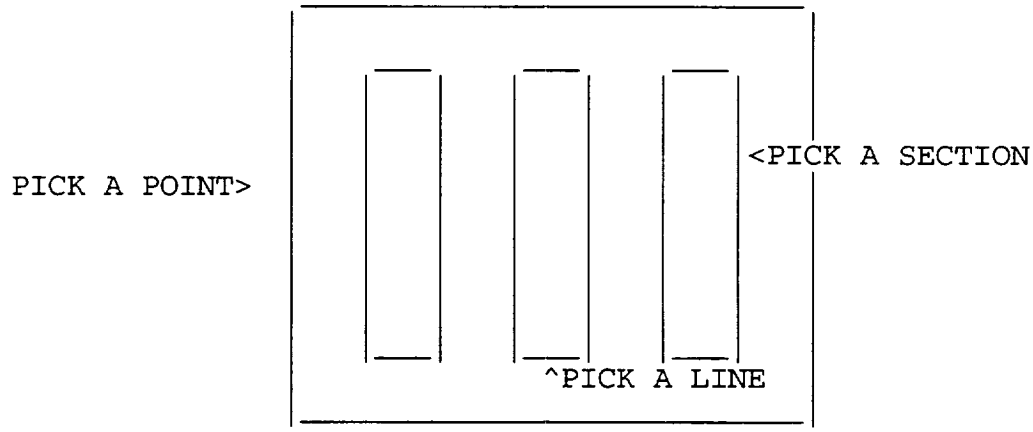
- Specify control points at the LE and TE
- Set the "SUBSET" number of points to that desired
- Set the "SUBSET" point spacing to that desired

3) Now run a program called "conv.f" which will mirror the upper surface ordinates from the "_____.crv" file as well as supply the wing surface grid program with the needed parameters to create the surface grid. The output file name to this program is "airfXXX.ord" (Note: XXX is the number of points describing the airfoil)

III. FOR S3D:

1) If this is the first time redistributing the airfoil points from sixseries.f then put the output sixseries file in the same format as the "airfXXX.ord" file above.
2) Once this is done run the surface grid program "WingSurf_.f which will give a first cut to the surface grid generation. Note, use the option of MG (multi-grid) when running the surface grid program, it will ask for this.
3) Now use the "Wingsurf_S3d.f" program which will take the upper surface of the wing only so that it can be read into S3d.
4) Its time to use S3d to redistribute the points at LE and TE. Note the following steps:

- Read in the file as unformatted MG Plot3d
- Swap indices so you can cluster at LE & TE Which can be done by going to[PGA] and selecting[SWAP INDICES]
- To select the section to be redistribute with the mouse making sure to be in the PICK MODE. Note, the mouse buttons give the following options:



Select the entire wing patch by using the right mouse button.

- Now redistribute the points by going to "GDP" and under this menu select "REDISTRIBUTE SECTION".
  - Specify the 1st and last spacing
  - Specify the # of points
  - Write out a file with the new distribution

- Remember now to swap back the indices

5) Now to put the new airfoil distribution in the proper format to read into the surface grid generator use the program "S3d_airf.f".
6) Finally, check the spacing with the program "sf.f" to make sure the stretching condition of sf=1.3 is met.

IV. GENERATING THE SURFACE GRID

Execute the surface grid generating program "WingSurf_new.f" to generate an output file which generates the wing surface.

NOTE: The following are inputs for the Surface Grid Generator:

- Taper Ratio or Aspect Ratio
- Leading Edge or Quarter chord sweep
- Number of spanwise points (cuts) on the wing
- Initial spacing in the spanwise direction at the tip chord.
- Final spacing in the spanwise direction at the root chord.
- Airfoil ordinate input file created from the above.

Finally, this will give an output file for the surface grid.

APPENDIX B

```
                        program WingSurf_new
c     ******************************************************************
c
c               Joseph A. Garcia
c               Date: Jan. 13, 1992
c
c     ******************************************************************
c      PROGRAM: This program will generate a surface grid for a
c               clipped delta wing with NACA 64A010 sections
c               using an Airfoil Potential Analytical Description
c
c      MODIFICATIONS:
c         MOD1:  To no longer use the Airfoil Description but to use as
c                an input from another code called sixseries.f the
c                normalized airfoil coordinates for a NACA 64A010, which
c                has been modified using Visual Grid (VG) to have
c                the desired chordwise point destribution. Also a span-
c                wise point distribution  which is develop by a program
c                name span_dist2.f and again modified by VG to have the
c                desired point spacing will now be an input to this code.
c
C           MOD2: THIS IS A MOD [10/5/91] TO EXTEND THE SWEEP INTO THE
C                 TIP SHAPE PORTION OF THE WING
c
c           MOD3: This modifies the code to allow for a taper ratio of
c                 one with equal leading edge and trailing edge sweeps
c                 that will now require a Aspect Ratio (AR) input.
c
c           MOD4: This mod will allow this surface grid generation code
c                 to be able to create any sweep clipped delta wing with
c                 out having to input a spanwise point destribution for
c                 each 1/4 sweep and taper ratio, instead the Vinokur
c                 stretching subroutine will be used to determine the
c                 distribution.
c
c           MOD5: This mod is to allow the user to either input sweep
c                 as either LE sweep or 1/4 chord sweep.
c
c           MOD6: This mod will allow this surface wing grid generation
c                 code to be able to create any sweep wing with an
c                 assigned aspect ratio "AR" which will sweep the trailing
c                 edge of the wing as necessary.
c
c           MOD7: This mod was done to have the "WingSurf_gen" give the
c                 TE_sweep for all the various wing inputs along with
c                 the span, AR, TR, LE_sweep, Qrt_sweep as necessary.
c
c           MOD8: This mod was done to sweep all of the tip zero section
c                 of the wing with the LE_sweep.
c
c           MOD9: This mod will cluster the zero thickness trailing edge
c                 points to match those of the swept wing.
c
c          MOD10: This mod will cluster the zero thickness Wint-Tip
                  section using the Vinokur streching routine and not just
                  mirroring the points off the wing.
```

```
c
c     INPUTS: Quarter-chord sweep angle (GAMMA), taper ratio (lamda)
c             surface grid dimensions (jmax,lmax), and normalized
c             airfoil ordinates file named "airfXXX.ord" (airf127.ord
c             or airf200.ord).
c
c     OUTPUT: PLOT3D-format surface grid of the wing
c
c     ****************************************************************
      parameter (jdim=500,kdim=100,ldim=10,idim=500)
      dimension x(jdim,kdim,ldim),y(jdim,kdim,ldim),z(jdim,kdim,ldim),
     +          x_U(idim), z_U(idim), x_L(idim), z_L(idim), yy(kdim),
     +          s(150), t(100),w(50),IDM(jdim),JDM(kdim),KDM(ldim)
       CHARACTER*30 OUTFILE,name,INFILE
 1000 FORMAT(A)
      REAL GAMA, lambda, t_10, t_11, t_12, t_13, t_14, t_15, t_21
     +        , X, t_22, t_23, t_24, t_25, Chord, span, sweep, y_edg
     +        , dely, delx, dely_t, delx_te, Chord_r, TE_length,Chord_t
     +        , yspan, d1, d2, stotin, LE_sweep, AR, LE_length, TE_sweep
     +        , Qrt_sweep, dt1, dt2, dt1t, dt2t, delwk
     +        , dw0w, dw1w, dw2w, dw3w, deltp2, dely_wt, thrdspan, sf
      INTEGER jmax, kmax, count, jmax_u, kmax_w, kmax_t, jmax_t
     +        , counter, jmax_te, jmax_te_U, npts_U, npts_L,tr_test1
     +        , lmax, llmax, kk, sw_type, AR_type, tr_test2, cont_test1
     +        , tmax, jj,MG,IGRID, wmax
c
c     Taper ratio = lambda
c  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
c  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
c       Qrt_sweep  = 1/4 chord sweep in DEGREES
c       GAMA       = 1/4 chord sweep in RADS
c       sweep      = Leading Edge sweep in RADS
c       LE_sweep   = Leading Edge sweep in DEGREES
c       TE_sweep   = Trailing Edge sweep in DEGREES
c       sweep_te   = Trailing Edge sweep in RADS
c       dt1t         = Initial TE Wake spacing @ tip
c       dt2t         = Final TE Wake spacing @ tip
c     sf          = Strecthing factor (1.3)
c  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
c  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
c
c  ----------   set default parameters   ----------
            sf = 1.3
         ngrid = 1
       Chord_r = 1.0
            d1 = 0.3
            d2 = 0.005
     TE_length = 0.5*Chord_r
c    ---------------------------------------------
c
      WRITE(*,'(a,$)')'If you KNOW what you want your TAPER RATIO to be
     +type "1" if NOT type "0": '
      read (*,*)tr_test1
          if(tr_test1 .eq. 1) then
            continue
              else
      WRITE(*,'(a,$)')'You must now specify a span since no taper was sp
     +ecified (.84): '
```

```
      read (*,*)span
                goto 1
                  endif
      WRITE(*,'(a,$)')'If the taper ratio is 1 type "1" or "0" if not:'
      read (*,*)tr_test2
          if(tr_test2 .eq. 1) then
               goto 2
                else
                  continue
                     endif
      WRITE(*,'(a,$)') ' INPUT taper ratio: '
      read (*,*) lambda
c
 1    PRINT*,'If you plan to specify Aspect Ratio type 1 or 0 if not:'
      read (*,*)AR_type
c
          IF(AR_type .eq. 1) THEN
      WRITE(*,'(a,$)')'INPUT Aspect Ratio desired normalized by root cho
     +rd: '
      read (*,*)AR
c  =====================================================
          if(tr_test1 .ne. 1 ) then
          lambda = (2*span/AR - 1.0 )
                     else
                        continue
                         endif
c  =====================================================
      WRITE(*,'(a,$)')'If Sweep is based on LE type "1" or "0" if 1/4C:'
      read (*,*)sw_type
          if(sw_type .eq. 1) then
      WRITE(*,'(a,$)') ' INPUT LE Sweep [deg]: '
      read (*,*) LE_sweep
       sweep= LE_sweep*(3.141592654/180)
       span = AR*(1+lambda)/2
       GAMA = ATAN( (span*TAN(sweep) + .25*(lambda - Chord_r))/span )
       Qrt_sweep = GAMA*(180/3.141592654)
       TE_sweep= ATAN( (span*TAN(sweep) - 1 + lambda)/span )
       TE_sweep= TE_sweep*(180/3.141592654)
c  =====================================================
          if(TE_sweep .lt. 0.0 ) then
      PRINT*,'YOUR CHOICE OF INPUT YEILDS A "-" TE_SWEEP'
      PRINT*,'AND THE BL CODE "WING" DOES NOT TAKE THIS'
      PRINT*,'SO IF YOU WANT TO CONTINUE ANYWAYS TYPE 1 else 0:'
      read(*,*) cont_test1
          if(cont_test1 .eq. 1 ) then
                      continue
                        else
      PRINT*,'  OK      !!!!!! TRY AGAIN  !!!!!!!!!!'
                        STOP
                          endif
                     else
                        continue
                         endif
c  =====================================================
                  else
      WRITE(*,'(a,$)') ' INPUT 1/4 Chord Sweep [deg]: '
      read (*,*) Qrt_sweep
       GAMA = Qrt_sweep*(3.141592654/180)
```

```
      span = AR*(1+lambda)/2
      sweep = ATAN( (span*TAN(GAMA) - .25*(lambda - Chord_r))/span )
      LE_sweep= sweep*(180/3.141592654)
      TE_sweep= ATAN( (span*TAN(sweep) - 1 + lambda)/span )
      TE_sweep= TE_sweep*(180/3.141592654)
                  endif
                ELSE
c  ---------------------------------------------------
     WRITE(*,'(a,$)')'If Sweep is based on LE type "1" or "0" if 1/4C:'
     read (*,*)sw_type
         if(sw_type .eq. 1) then
     WRITE(*,'(a,$)') ' INPUT LE Sweep [deg]: '
     read (*,*) LE_sweep
      sweep= LE_sweep*(3.141592654/180)
     WRITE(*,'(a,$)')'INPUT TE_sweep if Delta wing then use 0 deg: '
     read (*,*) TE_sweep
      TE_sweep= TE_sweep*(3.141592654/180)
c  ===================================================
        if(tr_test1 .ne. 1 ) then
          lambda = span*( TAN(TE_sweep) - TAN(sweep) ) + Chord_r
           if(lambda .lt. 0.0) then
     PRINT*,'YOU CHOSEN TO LARGE A SPAN FOR THESE SWEEPS'
           span = (0.0 - Chord_r)/( TAN(TE_sweep) -
    +                                TAN(sweep) )
     PRINT*,'SPAN MUST BE = or > ',span
     PRINT*,'         !!!!!! TRY AGAIN  !!!!!!!!!!'
                  STOP
                    else
                    continue
                      endif
                else
                    continue
                     endif
c  ===================================================
      span = (lambda - Chord_r)/(TAN(TE_sweep) - TAN(sweep))
      GAMA = ATAN( (span*TAN(sweep) + .25*(lambda - Chord_r))/span )
      Qrt_sweep= GAMA*(180/3.141592654)
      AR = 2*span/(1+lambda)
      TE_sweep= TE_sweep*(180/3.141592654)
                  else
     WRITE(*,'(a,$)') ' INPUT 1/4 Chord Sweep [deg]: '
     read (*,*) Qrt_sweep
      GAMA = Qrt_sweep*(3.141592654/180)
     WRITE(*,'(a,$)')'INPUT TE_sweep if Delta wing then use 0 deg: '
     read (*,*) TE_sweep
      TE_sweep= TE_sweep*(180/3.141592654)
c  ===================================================
        if(tr_test1 .ne. 1 ) then
          lambda = span*( TAN(TE_sweep) - TAN(sweep) ) + Chord_r
                   else
                      continue
                       endif
c  ===================================================
       if( TE_sweep .eq. 0.0) then
      span = (0.75*(1 - lambda))/TAN(GAMA)
      sweep= ATAN( (span*TAN(TE_sweep) + 1 - lambda)/span )
      LE_sweep= sweep*(180/3.141592654)
      TE_sweep= TE_sweep*(180/3.141592654)
```

```
        AR = 2*span/(1+lambda)
                  else
       sweep = (.25*TAN(TE_sweep) - TAN(GAMA))/(-.75)
       span= (0.25*(lambda - Chord_r) )/( (TAN(GAMA) - TAN(sweep)) )
c      sweep=ATAN( (span*TAN(GAMA) - .25*(lambda - Chord_r))/span)
       LE_sweep= sweep*(180/3.141592654)
       TE_sweep= TE_sweep*(180/3.141592654)
       AR = 2*span/(1+lambda)
                    endif
                      endif
                    ENDIF
c
        PRINT *, 'span= ', span
        PRINT *, 'LE_sweep= ', LE_sweep
        PRINT *, 'TE_sweep= ', TE_sweep
        PRINT *, 'Qrt_sweep= ', Qrt_sweep
        PRINT *, 'AR= ', AR
        PRINT *, 'Taper ratio= ', lambda
c
       WRITE(*,'(a,$)')'INPUT how many point in the spanwise [25]: '
       read (*,*)kmax_w
       WRITE(*,'(a,$)')'INPUT initial spacing in spanwise dir. [.05]: '
       read (*,*)d1
       WRITE(*,'(a,$)')'INPUT final spacing in the spanwise dir[.005]: '
       read (*,*)d2
c
c      ######################################################################
         CALL vinokur(s,kmax_w,span,d1,d2)
c      ######################################################################
          i=0
          do 4 i=1,kmax_w
        yy(i) = s(i)
          k = k + 1
        if(ABS(yy(i) - span) .lt. 0.001) kmax_w = k
 4       continue
c      ######################################################################
c             This section will set the spanwise outer boundary
c             for the tip zero section.
c      ######################################################################
c             ==============================
c                        MOD10
c             ==============================
             dely_wt = (yy(kmax_w) - yy(kmax_w-1) )*Chord_r
             dw1w = (yy(kmax_w) - yy(kmax_w-1) )*Chord_r
       Print*,' dely_wt= ',dely_wt
             dw3w = 0.
               wmax = 1
              dw0 = dw1w
             dw2w = 0.
c2/25/93     thrdspan = 0.3*span
             thrdspan = 1.0*span
        do 15 jj = 1,100
           deltp2 = .20*thrdspan
c           if(dw2w .lt. deltp2) then
            if(dw3w .lt. thrdspan) then
             dw0 = dw0*sf
                 wmax = wmax + 1
             dw3w = dw0
```

```
              dw2w = dw3w - dw3w/sf
                   else
                   continue
                     endif
 15    Continue
              kmax = kmax_w + (wmax -1)
c     -------------------------------------------
c        if(yy(i) .le. .5*span) then
c            kmax = kmax_w + (kmax_w - k)
c         print *,'kmax= ' ,kmax
c             endif
cc          print *,'kmax_w= ' ,kmax_w
             goto 3
c       ######################################################################
c
c       \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
c       This will open the spanwise ordinate data file ceated
c       and then read it into an array
c
c       open(21,file='span2.crv',status='old',form='formatted')
c        read(21,*)
c        read(21,*) kmax
c         k = 0
c          do 4 i=1,kmax
c        read(21,*) yy(i)
c          k = k + 1
c        if(ABS(yy(i) - span) .lt. 0.001) kmax_w = k
c         print *,'kmax_w= ' ,kmax_w
c4       continue
c          goto 3
c       \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
c
c
 2     WRITE(*,'(a,$)') ' INPUT Sweep [deg]: '
       read (*,*)sweep
        GAMA = sweep*(3.141592654/180)
        LE_sweep = sweep
        Qrt_sweep = sweep
        TE_sweep = sweep
        lambda = 1.0
c
c      WRITE(*,'(a,$)') ' INPUT LE or TE length [y/Cr] : '
c      read (*,*)LE_length
c       span = LE_length*COS(GAMA)
c
       WRITE(*,'(a,$)') ' INPUT Aspect Ratio normalized by root Chord: '
       read (*,*)AR
        span = AR*(1 + lambda)/2.0
        AR = 2*span/(1+lambda)
        PRINT *, 'span= ', span
        PRINT *, 'LE_sweep= ', LE_sweep
        PRINT *, 'Qrt_sweep= ', Qrt_sweep
        PRINT *, 'TE_sweep= ', TE_sweep
        PRINT *, 'AR= ', AR
       WRITE(*,'(a,$)')'INPUT how many point in the spanwise [kmax_w]: '
       read (*,*)kmax_w
       WRITE(*,'(a,$)') ' INPUT initial spacing in the spanwise dir. : '
       read (*,*)d1
```

```
      WRITE(*,'(a,$)') ' INPUT final spacing in the spanwise dir. : '
      read (*,*)d2
c
c      ######################################################################
         CALL vinokur(s,kmax_w,span,d1,d2)
c      ######################################################################
            k = 0
          do 6 i=1,kmax_w
        yy(i) = s(i)
          k = k + 1
        if(ABS(yy(i) - span) .lt. 0.001) kmax_w = k
        if(yy(i) .le. .5*span) then
             kmax = kmax_w + (kmax_w - k)
         print *,'kmax= ' ,kmax
          endif
c        print *,'kmax_w= ' ,kmax_w
 6       continue
c      ######################################################################
c         kmax_w = 0.75*kmax
c         yspan = span/(kmax_w-1)
c         do 2 i =1,kmax
c             yy(i) = yspan*(i-1)
c2      continue
c
c      \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
c      This will open the airfoil ordinate data file ceated
c      by the SIXSERIES code ref ___ and then read it into an array
c
c3     open(20,file='airf.ord',status='old',form='formatted')
 3     WRITE(*,'(a,$)')' ENTER grid AIRFOIL ORDINATE INFILE NAME:  '
        READ(*,1000)infile
        open(20,file=infile,status='old',form='formatted')
        read(20,1000) name
        read(20,*) npts_U
        read(20,*) (x_U(i),z_U(i),i=1,npts_U)
        read(20,*) npts_L
        read(20,*) (x_L(i),z_L(i),i=1,npts_L)
        read(20,*) jmax_te_U
        read(20,*) delx_te
        read(20,*)TE_length
c
c      \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
c
c
        jmax = npts_U + (npts_L-1)
      PRINT*,'HEY!!! jmax = ',jmax
        jmax_U = npts_U
        llmax = 1
c       dely = span/(.6*kmax-1)
c       kmax_w = 0.6*kmax
c       kmax_t = kmax - kmax_w
c
c       kmax = 1.2*kmax_w
c  ==============================
c           MOD9a
c  ==============================
      do 50 k=1,kmax_w
c  ==============================
```

```
      PRINT*,' k= ',k
c
c  ================================================================
c   This will add a zero thick section behind the "Wing-Trailing Edge"
c                ****  for the upper surface ****
c     MOD9:  Starting from the Tip of the wing
c  ================================================================
c
      TE_sweep = ATAN( (span*TAN(sweep)-1 + lambda)/span )
c     PRINT*,'********** y(j,k,1)= ',y(j,k,1)
c  ==============================
c           MOD9
c  ==============================
      Chord = (1 + yy(k)*(TAN(TE_sweep) - TAN(sweep)) )
      Chord_t = (1 + yy(kmax_w)*(TAN(TE_sweep) - TAN(sweep)) )
c      PRINT*,' Chord= ',Chord
c      PRINT*,' Chord_t= ',Chord_t
        delx_te = ( x_U(npts_U) - x_U(npts_U-1) )*Chord
        dt1t = ( x_U(npts_U) - x_U(npts_U-1) )*Chord_t
       PRINT*,' delx_te= ',delx_te
           dt1 = delx_te
        dt0 = dt1
            IF ( k .eq. 1) THEN
c         PRINT*,' HI 1!!'
        tmax = 1
        dt0 = dt1t
       dt2t = 0.
      do 7 j = 1,100
c   ***************************************************
c    NOTE: This is sometimes change to avoid certain
c          conditions in Vinokur subroutine that distorts
c          the grid spacing.
c   ***************************************************
c                 delwk = 0.12*TE_length
                  delwk = 0.13*TE_length
c   ***************************************************
c          PRINT*,' HI 2!! delwk= ',delwk
           if( dt2t .lt. delwk) then
c          PRINT*,' HI 3!!'
          dt0 = dt0*sf
              tmax = tmax + 1
            dt3t = dt0
            dt2t = dt3t - dt3t/sf
c      PRINT*,'#1  dt1t=',dt1t,' dt2t=',dt2t
             else
               continue
                  endif
 7      continue
           dt2t = dt3t - dt3t/sf
c      PRINT*,'#1  dt1t=',dt1t,' dt2t=',dt2t
c
             ELSE
               CONTINUE
                         ENDIF
c2/93        dt2t = dt3t - dt3t/sf
           dt2 = dt2t
c      PRINT*,'#2  dt1=',dt1,' dt2=',dt2
c
```

```
              jmax_te_U = tmax - 1
                  jmax_te = 2*jmax_te_U
c      PRINT*,'HEY 1 !!! jmax_te= ',jmax_te
c       PRINT*,'dt3t= ',dt3t
        PRINT*,'tmax= ',tmax
c         ################################
          CALL vinokur(t,tmax,TE_length,dt1,dt2)
c         ################################
            jj = tmax + 1
          if(tr_test2 .eq. 1) then
          TE_sweep = GAMA
              else
                continue
                   endif
       do 10 j= 1,jmax_te_U
          jj = jj - 1
c       PRINT*,'t(jj)= ',t(jj),'   jj= ',jj
           y(j,k,1) = yy(k)
       x(j,k,1) = Chord_r + y(j,k,1)*(TAN(TE_sweep)) + t(jj)
c       PRINT*,' x(j,k,1)= ',x(j,k,1),'   j= ',j
       z(j,k,1) = 0.0
 10        Continue
c
c   ================================================
c    This will compute the upper surface of the wing
c    starting from the root trailing edge.
c          ****  for the upper surface ****
c   ================================================
c
c
c   ==============================
c             MOD9
c   ==============================
c              i = npts_U - jmax_te_U + 1
               i = npts_U + 1
          do 20 j=jmax_te_U + 1,jmax_U + jmax_te_U
c   ==============================
               i =i - 1
           y(j,k,1) = yy(k)
              if(tr_test2 .eq. 1) then
                 Chord = 1.0
             x(j,k,1) = Chord * x_U(i) +  (y(j,k,1)*TAN(GAMA))
                  else
       TE_sweep = ATAN( (span*TAN(sweep)-1 + lambda)/span )
       Chord = (1 + y(j,k,1)*(TAN(TE_sweep) - TAN(sweep)) )
             x(j,k,1) = Chord * x_U(i) + y(j,k,1)*TAN(sweep)
                    endif
c
c       \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
         z(j,k,1) = Chord * z_U(i)
c       \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
 20             continue
c   ================================================
c    This will compute the lower surface of the wing
c    starting from the root leading edge
c   ================================================
         count= 1
c   ==============================
```

```
c           MOD9
c  ==============================
c       do 30 j=jmax_U+1,jmax - jmax_te_U
        do 30 j=jmax_U + jmax_te_U + 1,jmax + jmax_te_U
c  ==============================
             count = count + 1
          y(j,k,1) = yy(k)
c      \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
             if(tr_test2 .eq. 1) then
            x(j,k,1) = Chord * x_L(count) +  (y(j,k,1)*TAN(GAMA))
                 else
       TE_sweep = ATAN( (span*TAN(sweep)-1 + lambda)/span )
       Chord = (1 + y(j,k,1)*(TAN(TE_sweep) - TAN(sweep)) )
            x(j,k,1) = Chord * x_L(count) + y(j,k,1)*TAN(sweep)
                 endif
c      \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
c            This section will read in the ordinate of the airfiol and
c            convert it to the proper values to define the wing
c      \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
        z(j,k,1) = Chord * z_L(count)
c      \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
 30            continue
c
c  ===================================================================
c   This will add a zero thick section behind the "Wing - Trailing Edge"
c         ****  for the lower surface ****
c  ===================================================================
c  ==============================
c           MOD9
c  ==============================
            jj = 1
c           do 40 j= jmax - jmax_te_U + 1,jmax
            do 40 j= jmax + jmax_te_U + 1,jmax + 2*jmax_te_U
              jj = jj + 1
          y(j,k,1) = yy(k)
             if(tr_test2 .eq. 1) then
           x(j,k,1) = t(jj) + y(j,k,1)*TAN(GAMA)
                 else
       TE_sweep = ATAN( (span*TAN(sweep)-1 + lambda)/span )
       x(j,k,1) = Chord_r + y(j,k,1)*TAN(TE_sweep) + t(jj)
                   endif
            z(j,k,1) = 0.0
 40       Continue
c
c     ============================================
 50            continue
c     ============================================
c
c             kk = kmax_w  *** MOD10 ***
              kk = 1
       do 100 k= kmax_w+1, kmax
c  ===================================================================
c   This will add a zero thick section cff the "Wing Tip-Trailing Edge"
c         ****  for the upper surface ****
c  ===================================================================
c  ==============================
c           MOD10
c            kk = kk -1
```

```
c  ==============================
              dely_wt = (y(1,kmax_w,1) - y(1,kmax_w-1,1) )*Chord_r
              dw1w = (y(1,kmax_w,1) - y(1,kmax_w-1,1) )*Chord_r
        Print*,' dely_wt= ',dely_wt
                wmax = 1
               dw0 = dw1w
              dw3w = 0.
              dw2w = 0.
c2/25/93     thrdspan = 0.3*span
             thrdspan = 1.0*span
          do 55 jj = 1,100
             deltp2 = .2*thrdspan
c             if(dw2w .lt. deltp2) then
              if(dw3w .lt. thrdspan) then
               dw0 = dw0*sf
                   wmax = wmax + 1
               dw3w = dw0
               dw2w = dw3w - dw3w/sf
                    else
                    continue
                      endif
 55     Continue
              dw2w = dw3w - dw3w/sf
              dt1 = dely_wt
c             dt2 = deltp2
              dt2 = dw2w
c          ##################################
           CALL vinokur(w,wmax,thrdspan,dt1,dt2)
c          ##################################
           kk = kk + 1
c  ==============================
c
c  ==============================
c            MOD9
c  ==============================
c              i = npts_U  + 1
            jj = tmax + 1
c  ==============================
             do 60 j= 1,jmax_te_U
                 jj = jj - 1
c    +++++++++++++++ MOD10 ++++++++++++++++++
c         y(j,k,1) = yy(kmax_w) + (yy(kmax_w) - yy(kk))
          y(j,k,1) = w(kk) + yy(kmax_w)
c    ++++++++++++++++++++++++++++++++++++++++
             IF(tr_test2 .eq. 1) THEN
c  ==============================
c            MOD9
c  ==============================
c     x(j,k,1)=x_U(i) + y(j,k,1)*TAN(GAMA)
      x(j,k,1)=x(j,kmax_w,1) + (y(j,k,1) - y(j,kmax_w,1))*TAN(GAMA)
c  ==============================
                              ELSE
      TE_sweep = ATAN( (span*TAN(sweep) - 1 + lambda)/span )
      Chord_t = (1 + y(j,kmax_w+1,1)*(TAN(TE_sweep) - TAN(sweep)) )
C     ****************************************************************
C     ************************MOD8************************************
C             THIS IS A MOD TO EXTEND THE LE_SWEEP INTO
C                   THIS ZERO THICKNESS SECITON
```

```
C     ****************************************************************
C  ==============================
C           MOD9
C  ==============================
C     x(j,k,1)=Chord_t + x_U(i) + y(j,k,1)*TAN(sweep) - Chord_r
      x(j,k,1)=t(jj) + Chord_t*x_U(npts_U) + y(j,k,1)*TAN(sweep)
C  ==============================
C     ****************************************************************
                                ENDIF
C
           z(j,k,1) = 0.0
 60        Continue
C  ========================================================
C   This will add a zero thickness section off the "Wing Tip" chord
C     ********   For the Upper surface *******
C  ========================================================
C
C  ==============================
C           MOD9
C  ==============================
C                i= npts_U - jmax_te_U  + 1
             i = npts_U + 1
C       do 70 j= jmax_te_U + 1,jmax_U
        do 70 j= jmax_te_U + 1,jmax_U + jmax_te_U
C  ==============================
             i = i - 1
C
C    +++++++++++++++ MOD10 ++++++++++++++++++
C         y(j,k,1) = yy(kmax_w) + (yy(kmax_w) - yy(kk))
          y(j,k,1) = w(kk) + yy(kmax_w)
C    ++++++++++++++++++++++++++++++++++++++++
C
             IF(tr_test2 .eq. 1) THEN
                Chord_t = 1.0
            x(j,k,1) = Chord_t * x_U(i) +  (y(j,k,1)*TAN(GAMA))
                          ELSE
      TE_sweep = ATAN( (span*TAN(sweep)-1 + lambda)/span )
      Chord_t = (1 + y(j,kmax_w+1,1)*(TAN(TE_sweep) - TAN(sweep)) )
C     ****************************************************************
C     ************************MOD8************************************
C            THIS IS A MOD TO EXTEND THE LE_SWEEP INTO
C                  THIS ZERO THICKNESS SECITON
C     ****************************************************************
      x(j,k,1)= Chord_t * x_U(i) + y(j,k,1)*TAN(sweep)
C     ****************************************************************
                                ENDIF
C      $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
        z(j,k,1) = 0.0
C      $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
 70           continue
C
C  ========================================================
C   This will add a zero thickness section off the "Wing tip" chord
C     ********   For the Lower surface *******
C  ========================================================
C
       counter = 1
C  ==============================
```

```
c           MOD9
c  ==============================
c       do 80 j= jmax_U + 1,jmax - jmax_te_U
        do 80 j= jmax_U + jmax_te_U + 1,jmax + jmax_te_U
c  ==============================
        counter = counter + 1
c     +++++++++++++++ MOD10 ++++++++++++++++++
c          y(j,k,1) = yy(kmax_w) + (yy(kmax_w) - yy(kk))
           y(j,k,1) = w(kk) + yy(kmax_w)
c     ++++++++++++++++++++++++++++++++++++++++
c
              IF(tr_test2 .eq. 1) THEN
                 Chord_t = 1.0
           x(j,k,1) = Chord_t * x_L(counter) +  (y(j,k,1)*TAN(GAMA))
                           ELSE
       TE_sweep = ATAN( (span*TAN(sweep)-1 + lambda)/span )
       Chord_t = (1 + y(j,kmax_w+1,1)*(TAN(TE_sweep) - TAN(sweep)) )
c
C      ***********************************************************************
C      ************************MOD8***********************************************
C             THIS IS A MOD TO EXTEND THE LE_SWEEP INTO
C                   THIS ZERO THICKNESS SECITON
C      ***********************************************************************
        x(j,k,1)= Chord_t * x_L(counter) + y(j,k,1)*TAN(sweep)
C      ***********************************************************************
                                  ENDIF
C       $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
         z(j,k,1) = 0.0
 80            continue
c
c  ============================================================
c  This will add a zero thick section off the "Wing Tip- Trailing Edge"
c           ****  for the lower surface ****
c  ============================================================
c
c  ==============================
c               MOD9
c  ==============================
c                   i = npts_L - jmax_te_U
                   jj = 1
c          do 90 j= jmax - jmax_te_U + 1, jmax
           do 90 j= jmax + jmax_te_U + 1, jmax + jmax_te
c                   i = i + 1
                   jj = jj + 1
c     +++++++++++++++ MOD10 ++++++++++++++++++
c          y(j,k,1) = yy(kmax_w) + (yy(kmax_w) - yy(kk))
           y(j,k,1) = w(kk) + yy(kmax_w)
c     ++++++++++++++++++++++++++++++++++++++++
c
              IF(tr_test2 .eq. 1) THEN
c      x(j,k,1)=x_L(i) + y(j,k,1)*TAN(GAMA)
       x(j,k,1)=x(j,kmax_w,1) + (y(j,k,1) - y(j,kmax_w,1))*TAN(GAMA)
                           ELSE
       TE_sweep = ATAN( (span*TAN(sweep)-1 + lambda)/span )
       Chord_t = (1 + y(j,kmax_w+1,1)*(TAN(TE_sweep) - TAN(sweep)) )
C      ***********************************************************************
C      ************************MOD8***********************************************
C             THIS IS A MOD TO EXTEND THE LE_SWEEP INTO
```

```
C                    THIS ZERO THICKNESS SECITON
C     ***************************************************************
c     x(j,k,1)=Chord_t + x_L(i) + y(j,k,1)*TAN(sweep) - Chord_r
      x(j,k,1)=t(jj) + Chord_t*x_U(npts_U) + y(j,k,1)*TAN(sweep)
C     ***************************************************************
                                  ENDIF
c
           z(j,k,1) = 0.0
 90        Continue
c     ===========================================
 100            continue
c     ===========================================
c
c     write grid
c
       WRITE(*,'(a,$)')' ENTER grid FILE NAME:  '
        READ(*,1000)outfile
       WRITE(*,'(a,$)')' IF YOU WANT A MULTI GRID OUTPUT TYPE 1:  '
      READ(*,*)MG
c
c     change 'binary' to 'unformatted' to run on CRAY 2 or VAX
c
      OPEN(UNIT=7,FILE=outfile,STATUS='new',
     +     form='unformatted')
c  ==============================
c           MOD9
c  ==============================
      PRINT*,'HEY 2 !!! jmax_te= ',jmax_te
      jmax = jmax + jmax_te
      PRINT*,'HEY 2 !!! jmax= ',jmax
c  ==============================
           IDM(1) = jmax
         JDM(1) = kmax
         KDM(1) = llmax
       IF(MG .ne. 1) THEN
c    ***************************************************************
      WRITE(7) jmax,kmax,llmax
      WRITE(7)   (((X(J,K,L), J=jmax,1,-1), K=1,kmax), L=1,llmax),
     +           (((Y(J,K,L), J=jmax,1,-1), K=1,kmax), L=1,llmax),
     +           (((Z(J,K,L), J=jmax,1,-1), K=1,kmax), L=1,llmax)

           ELSE

           NGRID = 1
      WRITE(7) NGRID
      WRITE(7)  (IDM(IGRID),JDM(IGRID),KDM(IGRID),IGRID=1,NGRID)
      DO 110 IGRID= 1,NGRID
      WRITE(7)
     + (((X(I,J,K),
     +    I=IDM(IGRID),1,-1),J=1,JDM(IGRID)),K=1,KDM(IGRID)),
     + (((Y(I,J,K),
     +    I=IDM(IGRID),1,-1),J=1,JDM(IGRID)),K=1,KDM(IGRID)),
     + (((Z(I,J,K),
     +    I=IDM(IGRID),1,-1),J=1,JDM(IGRID)),K=1,KDM(IGRID))
 110     CONTINUE
c
            ENDIF
        stop
```

```
        end
c     ***********************************************************************
c
c
        subroutine vinokur(s,lmax,smax,ds1e,ds2e)
c
c  stretches points on a surface so that a specified spacing
c  at the boundaries is satisfied. Taken from NASA CR 3313 by
c  Vinokur (1980).
c
c  In this version, 4 distinct iterations are made to better
c  match the resulting delta-s values to the requested values.
c  The four iterations are summarized below:
c
c  1.  delta-s is set equal to the desired value.
c  2.  delta-s from the last iteration is corrected from a Taylor
c      series expansion.
c  3.  delta-s is calculated from a linear fit between the first two
c      guesses.
c  4.  delta-s is calculated from a quadratic fit between the first
c      three guesses, if indeed a quadratic will pass through the
c      desired value.  If it doesn't, it takes the value calculated
c      after three swipes.
c
c  Additionally, this version uses the approximate inverse solution
c  for y=sin(x)/x and y=sinh(x)/x rather than a Newton iteration.  The
c  approximate solution was also taken from NASA CR 3313.
c
c
      common /io/ input,kopy,default
      dimension s(200), d1(4,2),d2(4,2)
c
c--------------------------------
c  for an IRIS 2500,
      emax=87.0
c--------------------------------
c
      dsavg=smax/float(lmax-1)
c 21   write(*,103)dsavg
c       PRINT*,'ds1e= ',ds1e
c       PRINT*,'ds2e= ',ds2e
c       PRINT*,'smax= ',smax
21    dsavg=0.001
c      ds1e=dsavg
c      call realval(1,1,ds1e,q,q,*21,*101)
      if(ds1e.ge.       smax.or.ds1e.lt. 0.0)go to 21
22    dsavg=0.01
c22   write(*,104)dsavg
c      ds2e=dsavg
c      call realval(1,1,ds2e,q,q,*22,*101)
      if(ds2e.ge.(smax-ds1e).or.ds2e.lt. 0.0)go to 21
      if(ds1e.eq.0.0.and.ds2e.eq.0.0)then
         kase=0
         ds1e=dsavg
         ds2e=dsavg
         nlast=4
      else if(ds1e.eq.0.0)then
         kase=1
```

```
         nlast=1
c 23      write(6,106)
23       continue
c         call realval(0,1,slop,no,no,*23,*101)
         if(slop.lt.0.0.or.slop.gt.1.0)go to 23
         ds1e=-slop
      else if(ds2e.eq.0.0)then
         kase=2
         nlast=1
c 24      write(6,106)
24       continue
c         call realval(0,1,slop,no,no,*24,*101)
         if(slop.lt.0.0.or.slop.gt.1.0)go to 24
         ds2e=-slop
      else
         kase=0
         nlast=4
      end if
      dss1=0.0
      dss2=0.0
c
      do 6 n=1,nlast
      if(n.le.2)then
         ds1=ds1e-0.5*dss1
         ds2=ds2e+0.5*dss2
         d1(n,1)=ds1
         d2(n,1)=ds2
c       PRINT*,'d1(1,1)= ',d1(1,1)
c       PRINT*,'d1(1,2)= ',d1(1,2)
c       PRINT*,'d1(2,1)= ',d1(2,1)
c       PRINT*,'d1(2,2)= ',d1(2,2)
      else if(n.eq.3)then
         ds1=-d1(1,2)*(d1(2,1)-d1(1,1))/(d1(2,2)-d1(1,2))+d1(1,1)
c       PRINT*,'d2(1,1)= ',d2(1,1)
c       PRINT*,'d2(1,2)= ',d2(1,2)
c       PRINT*,'d2(2,1)= ',d2(2,1)
c       PRINT*,'d2(2,2)= ',d2(2,2)
         ds2=-d2(1,2)*(d2(2,1)-d2(1,1))/(d2(2,2)-d2(1,2))+d2(1,1)
c       PRINT*,'ds1= ',ds1
c       PRINT*,'ds2= ',ds2
c       PRINT*,'nlast= ',nlast
c       PRINT*,'HELP!!!'
         if(ds1.lt.0.0)ds1=0.5*amin1(d1(1,1),d1(2,1))
         if(ds2.lt.0.0)ds2=0.5*amin1(d2(1,1),d2(2,1))
         d1(n,1)=ds1
         d2(n,1)=ds2
      else if(n.eq.4)then
         denom=-(d1(1,1)-d1(2,1))*(d1(2,1)-d1(3,1))*(d1(3,1)-d1(1,1))
         a11=d1(2,1)-d1(3,1)
         a21=d1(3,1)**2-d1(2,1)**2
         a31=d1(2,1)*d1(3,1)*(d1(2,1)-d1(3,1))
         a12=d1(3,1)-d1(1,1)
         a22=d1(1,1)**2-d1(3,1)**2
         a32=d1(3,1)*d1(1,1)*(d1(3,1)-d1(1,1))
         a13=d1(1,1)-d1(2,1)
         a23=d1(2,1)**2-d1(1,1)**2
         a33=d1(1,1)*d1(2,1)*(d1(1,1)-d1(2,1))
         b1=(a11*d1(1,2)+a12*d1(2,2)+a13*d1(3,2))/denom
```

```
         b2=(a21*d1(1,2)+a22*d1(2,2)+a23*d1(3,2))/denom
         b3=(a31*d1(1,2)+a32*d1(2,2)+a33*d1(3,2))/denom
         disc=(b2*b2-4.*b1*b3)
         if(disc.lt.0.0)go to 8
         dd1=(-b2+sqrt(disc))/(2.*b1)
         dd2=(-b2-sqrt(disc))/(2.*b1)
         dd3=d1(3,1)
         if(abs(dd1-dd3).lt.abs(dd2-dd3))then
            ds1=dd1
         else
            ds1=dd2
         end if
         denom=-(d2(1,1)-d2(2,1))*(d2(2,1)-d2(3,1))*(d2(3,1)-d2(1,1))
         a11=d2(2,1)-d2(3,1)
         a21=d2(3,1)**2-d2(2,1)**2
         a31=d2(2,1)*d2(3,1)*(d2(2,1)-d2(3,1))
         a12=d2(3,1)-d2(1,1)
         a22=d2(1,1)**2-d2(3,1)**2
         a32=d2(3,1)*d2(1,1)*(d2(3,1)-d2(1,1))
         a13=d2(1,1)-d2(2,1)
         a23=d2(2,1)**2-d2(1,1)**2
         a33=d2(1,1)*d2(2,1)*(d2(1,1)-d2(2,1))
         b1=(a11*d2(1,2)+a12*d2(2,2)+a13*d2(3,2))/denom
         b2=(a21*d2(1,2)+a22*d2(2,2)+a23*d2(3,2))/denom
         b3=(a31*d2(1,2)+a32*d2(2,2)+a33*d2(3,2))/denom
         disc=(b2*b2-4.*b1*b3)
         if(disc.le.0.0)go to 8
         dd1=(-b2+sqrt(disc))/(2.*b1)
         dd2=(-b2-sqrt(disc))/(2.*b1)
         dd3=d2(3,1)
         if(abs(dd1-dd3).lt.abs(dd2-dd3))then
            ds2=dd1
         else
            ds2=dd2
         end if
         if(ds1.lt.0.0.or.ds2.lt.0.0)go to 8
      end if
c
c  calculate constants
      s0=smax/float(lmax-1)/ds1
      s1=smax/float(lmax-1)/ds2
      b=sqrt(s0*s1)
      a=sqrt(s0/s1)
      if(kase.eq.1)then
         b=s1
      else if(kase.eq.2)then
         b=s0
      end if
c
c  calculate x based on value of B
      if(b-1.)1,2,3
c
c  x is real
 1    if(b.lt.0.26938972)then
         pi=4.*atan(1.)
         x=pi*(1.    -b     +          b**2 - (1.+pi**2/6.)*b**3
     *    + 6.794732*b**4  -13.205501*b**5     + 11.726095*b**6)
      else
```

```
         c=1.-b
         x= sqrt(6.*c)*(1.
     *           +0.15*c    + 0.057321429*c**2 +0.048774238*c**3
     *    -0.053337753*c**4 + 0.075845134*c**5)
      end if
      go to 4
c
c  x is zero
 2    x=0.
      go to 4
c
c  x is imaginary
 3    if(b.lt.2.7829681)then
         c=b-1.
         x= sqrt(6.*c)*(1.
     *           -0.15*c    + 0.057321429*c**2 -0.024907295*c**3
     *   +0.0077424461*c**4 -0.0010794123*c**5)
      else
         v=alog(b)
         w=1./b - 0.028527431
         x= v + (1.+1./v)*alog(2.*v) -0.02041793
     *    + 0.24902722*w     +  1.9496443*w**2
     *    -  2.6294547*w**3  + 8.56795911*w**4
      end if
c
c  distribute points along boundary
c
 4    continue
      if(kase.eq.1.or.kase.eq.2)then
         s(1    ) = 0.0
         s(lmax) = smax
         do 9 i=2,lmax-1
              j= lmax+1-i
         xi=float(i-1)/(lmax-1)
         if(b.gt.1.0001)then
            u1=1. + tanh(x/2.*(xi-1.))/tanh(x/2.)
         else if(b.lt.0.9999)then
            u1=1. + tan (x/2.*(xi-1.))/tan (x/2.)
         else
            u1= xi*(1.-.5*(b-1.)*(1.-xi)*(2.-xi))
         end if
         u2=sinh(xi*x)/sinh(x)
         if(kase.eq.1)then
            fact=abs(ds1e)
            s(j) = ( (1.-fact)*(1.-u1) + fact*(1.-u2) ) *smax
         else if(kase.eq.2)then
            fact=abs(ds2e)
            s(i) = ( (1.-fact)*    u1   + fact*    u2   ) *smax
         end if
 9       continue
      else
         do 5 i=1,lmax
         xi=float(i-1)/float(lmax-1)
         cnum=x*(xi-0.5)
         cden=x/2.
         if(b.lt.0.9999)then
            cc=tan(cnum)/tan(cden)
            u=0.5*(1.+cc)
```

```
         else if(b.ge.0.9999.and.b.le.1.0001)then
            u=xi*(1.+2.*(b-1.)*(xi-0.5)*(1.-xi))
         else if(b.gt.1.0001)then
            cc=tanh(cnum)/tanh(cden)
            u=0.5*(1.+cc)
         end if
 5       s(i)=u*smax/(a+(1.-a)*u)
       end if
c
       if(lmax.ge.4)then
          dss1=(  -s(4)   +4.*s(3)      -5.*s(2)      +2.*s(1))
          dss2=(2.*s(lmax)-5.*s(lmax-1)+4.*s(lmax-2)   -s(lmax-3))/2.
       end if
c
       es1=s(2)-s(1)
       es2=s(lmax)-s(lmax-1)
       if(n.ne.4)then
          d1(n,2)=es1-ds1e
          d2(n,2)=es2-ds2e
       end if
 6     continue
c
 8     esmin= 1.0e+08
       esmax=-1.0e+08
       do 7 j=2,lmax
       stmp=s(j)-s(j-1)
       if(stmp.lt.esmin)then
          jnj=j
          esmin=stmp
       end if
       if(stmp.gt.esmax)then
          jxj=j
          esmax=stmp
       end if
 7     continue
c       write(6,105)es1,es2,jnj-1,jnj,esmin,jxj-1,jxj,esmax
c
 101   return
 103   format(/,6x,'enter delta s at beginning of arclength',
      *  /,6x,'(default = ',g12.5,' 0.= auto-spacing)',t59,'>',$)
 104   format(/,6x,'enter delta s at end of arclength',
      *  /,6x,'(default = ',g12.5,' 0.= auto-spacing)',t59,'>',$)
 105   format(/,6x,'computed spacing at beginning:',g12.5,/,
      *6x,          '                            end:',g12.5,/,
      *6x,          'minimum spacing   (i=',i3,',',i3,'):',g12.5,/,
      *6x,          'maximum spacing   (i=',i3,',',i3,'):',g12.5)
 106   format(6x,'   enter the degree of stretching',/,
      *       6x,'   (between 0. (tanh) and 1. (sinh) )'t59,'>',$)
       end
```

APPENDIX C

```
                                program  sf
c  ******************************************************
c
c           Joseph A. Garcia
c           Date:  Jan 1993
c
c  ******************************************************
c     This program will check the streching factor (sf) for a given
c       input of points and flag the user when the "sf" is larger
c           then the critical value of 1.3 in this case.
c
      parameter(ii=201,jj=201,kk=3)
      dimension xx(ii),yy(ii),del_x(ii),del_y(ii),del_r(ii)
      character*20 ident
      common /corner/ x1,y1l,y1u,xi,yil,yiu,x4,y4l,y4u,x5,y5l,y5u
c
c----read in x locals
1000  format(A)
       CHARACTER*30 infile
      WRITE(*,'(a,$)')' ENTER FILE NAME :  '
         READ(*,1000)infile
      open(30,file=infile, status='old',form='formatted')
      read(30,1000) ident
      read(30,*) idim
      write(*,*) 'idim ',idim
      read(30,*) (xx(i),yy(i),i=1,idim)
      write(*,*) xx(1)
      write(*,*) xx(2)
      do 20 i=1,idim
            del_x(i) = xx(i) - xx(i-1)
            del_y(i) = yy(i) - yy(i-1)
            del_r(i) = sqrt((del_x(i))**2 + (del_y(i))**2)
20    continue
      do 25 i=3,idim
c     sf = (xx(i)-xx(i-1))/(xx(i-1)-xx(i-2))
      sf = del_r(i)/del_r(i-1)
      if (sf.lt.1.0) then
        sf = 1.0/sf
        endif
      write(*,*) i,sf,xx(i)
        if (sf.gt.1.3) then
        write(*,*) '-------------------------'
        endif
25    continue
      stop
      end
```

```
                                Program airf_2dsurf
c      ***********************************************************
c           Joseph A. Garcia
c           Date: Jan 1993
c
c      ***********************************************************
c          This program will read the output of the sixseries code
c          and then create an upper surface curve of the airfoil
c          with a zero thickness trailing edge section to be
c          used as the 2d surface grid on VISUAL GRID for
c          redistribution.
c      ***********************************************************
       parameter(idim=200,jdim=200,kdim=5)
       dimension x_U(idim),z_U(jdim),x_te(100),z_te(100)
        integer npts, npts_0
         real delx_te
         character*20 name, infile, airfoil,wing
 1000    FORMAT(A)
        WRITE(*,'(a,$)') 'Enter the input file name :'
        READ(*,1000) infile
c       WRITE(*,'(a,$)') 'Enter the airfoil in consideration: '
c       READ(*,1000) airfoil
        WRITE(*,'(a,$)') 'Enter the number pts in the zero section:'
        READ(*,10) npts_0
 10      format(I3)
      open(25,file=infile,status='old',form='formatted')
      read(25,1000) name
      read(25,*) npts
      read(25,*) (x_U(i),z_U(i),i=1,npts)
c
        delx_te = .5/npts_0
c
c       WRITE(*,'(a,$)') 'Enter an output file name : '
c       READ(*,1000) wing
       open(26,file='airf.crv',status='old',form='formatted')
       WRITE(26,1000) name
       WRITE(26,40) npts + npts_0+1
       WRITE(26,50) (x_U(i),z_U(i),i=1,npts)
        do 20 i=1,npts_0 + 1
          x_te(i) = 1 + delx_te*(i-1)
          z_te(i) = 0.0
 20      continue
       WRITE(26,50) (x_te(i),z_te(i),i=1,npts_0+1)
 40      format(I4,1x,'Upper Coordinates')
 50      format(e14.8,3x,e14.8)
         stop
           END
```

```
                              Program S3d_airf
c      ********************************************************
c           By:   Joseph A. Garcia
c
c          This program will create airfoil ordinate file
c          "airf.ord" WingSurf generator from the first
c           cut WingSurf surface grid modified on S3d
c
c           Date: Jan 22, 1993
c      ********************************************************
        parameter(idim=200,jdim=200,kdim=5)
        dimension x_U(idim),z_U(jdim), x_te(20), z_te(20)
     +           ,IDM(5),JDM(5),KDM(5),X(idim,jdim,kdim)
     +           ,Y(idim,jdim,kdim),Z(idim,jdim,kdim)
        INTEGER npts, npts_0, ii,IGRID, form_test
        REAL delx_te, TE_lngth, delwk
        character*20 name,wing,infile,airfoil,outfile,formm
c   -----------------Defaults ----------------
c
        npts_0 = 25
c   ----------------------------------------
 1000    FORMAT(A)
       WRITE(*,'(a,$)') 'Enter the input file name :'
       READ(*,1000) infile
       WRITE(*,'(a,$)')'If file is fomatted type 1 or 0 if unform.:'
       READ(*,*)form_test
          if ( form_test .eq. 1)then
            formm = 'formatted'
             else
             formm = 'unformatted'
                 endif
         WRITE(*,'(a,$)') 'Enter the airfoil in consideration: '
         READ(*,1000) airfoil
         WRITE(*,'(a,$)')'What do you want to use as the TE sec lngth:'
         READ(*,10) TE_lngth
 10      FORMAT(f3.1)
c      PRINT*,'formm= ',formm
        open(7,file=infile,status='old',form=formm)
c
         IF ( form_test .eq. 1)then
       PRINT*,'FORMATTED'
         READ(7,*) NGRID
         READ(7,*) (IDM(IGRID),JDM(IGRID),KDM(IGRID),IGRID=1,NGRID)
         DO 15 IGRID= 1,NGRID
         READ(7,*)
     + (((X(I,J,K),
     +    I=1,IDM(IGRID)),J=1,JDM(IGRID)),K=1,KDM(IGRID)),
     + (((Y(I,J,K),
     +    I=1,IDM(IGRID)),J=1,JDM(IGRID)),K=1,KDM(IGRID)),
     + (((Z(I,J,K),
     +    I=1,IDM(IGRID)),J=1,JDM(IGRID)),K=1,KDM(IGRID))
 15      CONTINUE
            ELSE
       PRINT*,'UNFORMATTED'
         READ(7) NGRID
         READ(7) (IDM(IGRID),JDM(IGRID),KDM(IGRID),IGRID=1,NGRID)
         DO 20 IGRID= 1,NGRID
         READ(7)
```

```
     + (((X(I,J,K),
     +    I=1,IDM(IGRID)),J=1,JDM(IGRID)),K=1,KDM(IGRID)),
     + (((Y(I,J,K),
     +    I=1,IDM(IGRID)),J=1,JDM(IGRID)),K=1,KDM(IGRID)),
     + (((Z(I,J,K),
     +    I=1,IDM(IGRID)),J=1,JDM(IGRID)),K=1,KDM(IGRID))
 20     CONTINUE
           ENDIF
c
        WRITE(*,'(a,$)') 'Enter the output file name :'
        READ(*,1000) outfile
        open(26,file=outfile,status='new',form='formatted')
        WRITE(26,1000) airfoil
c   ------------------------------------------------------------
c   ------------------------------------------------------------
c     This section will put the plot3d mg coordinates
c       into the 1d airfoil ordinate format
c   ------------------------------------------------------------
c
              npts = IDM(1)
              ii = npts + 1
         do 30 i=1,npts
            ii = ii - 1
               j=1
               k=1
         x_U(i) =X(ii,j,k)
         z_U(i) =Z(ii,j,k)
 30      continue

c   ------------------------------------------------------------
c        Writing out the airfoil ordinates
c   ------------------------------------------------------------
        WRITE(26,40) npts
c       WRITE(26,50) (x_U(i),z_U(i),i=1,npts)
        WRITE(26,50) (x_U(i),-z_U(i),i=1,npts)
c      WRITE(25,50)
c     + (((X(I,J,K),
c     +    I=1,IDM(1)),J=1,JDM(1)),K=1,KDM(1)),
c     + (((Z(I,J,K),
c     +    I=1,IDM(1)),J=1,JDM(1)),K=1,KDM(1))
c       WRITE(26,55) (x_te(i),z_te(i),i=1,npts_0)
c   ------------------------------------------------------------
        WRITE(26,45) npts
c       WRITE(26,55) (x_U(i),-z_U(i),i=1,npts)
        WRITE(26,55) (x_U(i),z_U(i),i=1,npts)
c       WRITE(26,55) (x_te(i),-z_te(i),i=1,npts_0)
c   ------------------------------------------------------------
           delx_te = x_U(npts) - x_U(npts-1)
c   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
c      This section will estimate # pts needed in the
c      wing wake section
c           npts_0 = NINT(TE_lngth/delx_te)
c   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
               npts_0 = 0
               dt0 = delx_te
            dt2t = 0.
              delwk = 0.1*TE_lngth
     do 35 j = 1,100
```

```
            if( dt2t .lt. delwk) then
          dt0 = dt0*1.2
              npts_0 = npts_0 + 1
            dt3t = dt0
          dt2t = dt3t - dt3t/1.2
                 else
                  continue
                        endif
 35     continue
c    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
       WRITE(26,60) npts_0
       WRITE(26,65) delx_te
       WRITE(26,70) TE_lngth
 40     format(I4,1x,'Upper Coordinates')
 45     format(I4,1x,'Lower Coordinates')
 50     format(e14.8,3x,e14.8)
 55     format(e14.8,3x,e15.8)
 60      format(I4,1x,'= Number of pionts in the TE zero section')
 65      format(e14.8,1x,'= Delta X in the TE zero section')
 70      format(f3.1,1x,'= Length of the TE zero section')
         stop
           END
```

APPENDIX D

```
#! /bin/sh
JN='LRE60WING1_case1'
SN='COSAL_i18000'
MK='make4.eagle'
#  COMPILE THE CODES
#
if test -s Tran_rpt_n8.p3d.old
then
mv Tran_rpt_n8.p3d.old Tran_rpt_n8.p3d.older
fi
if test -s Tran_rpt_n8.p3d.new
then
mv Tran_rpt_n8.p3d.new Tran_rpt_n8.p3d.old
fi
cd /u0/rfa/jgarcia/stab_src_dir
make -f $MK
cp cosal_4.exe /u0/rfa/jgarcia/$JN/$SN
mv wing.exe /u0/rfa/jgarcia/$JN/$SN
mv stabin.exe /u0/rfa/jgarcia/$JN/$SN
mv getstab.exe /u0/rfa/jgarcia/$JN/$SN
cp interp_n5_8_p3d.exe /u0/rfa/jgarcia/$JN/$SN
mv plot3d_tran.exec /u0/rfa/jgarcia/$JN/$SN
cd /u0/rfa/jgarcia/$JN/$SN
chmod +x *.exe
mkdir job1
cd job1
#cp /u0/rfa/jgarcia/stab_run_dir/run* /u0/rfa/jgarcia/$JN/$SN
cp /u0/rfa/jgarcia/stab_run_dir/run*
/u0/rfa/jgarcia/$JN/$SN/job1
if test -s Tran_frnt.p3d.old
then
mv Tran_frnt.p3d.old Tran_frnt.p3d.older
fi
if test -s Tran_frnt.p3d.new
then
mv Tran_frnt.p3d.new Tran_frnt.p3d.old
fi
if test -s Tran_rpt_n8.p3d.old
then
mv Tran_rpt_n8.p3d.old Tran_rpt_n8.p3d.older
fi
if test -s Tran_rpt_n8.p3d.new
then
mv Tran_rpt_n8.p3d.new Tran_rpt_n8.p3d.old
fi
mv cosal.time cosal.time.old
#
#    THIS IS WHERE THE LOOP FOR THE SPECIFIED SPAN STATION
STARTS
#
#for case in fort.73 fort.74 fort.75 fort.76 fort.77 fort.78
fort.79 fort.80 fort.81 fort.82 fort.83 fort.84 fort.85
```

```
fort.86 fort.87 fort.88 fort.89 fort.90 fort.91 fort.92
fort.93 fort.94 fort.95 fort.96
#
#for case in fort.74 fort.75 fort.77 fort.79
#
for case in fort.74 fort.75 fort.77 fort.79 fort.81 fort.83
fort.85 fort.87
#
do
# CLEAN UP THE OUTPUT FILES
#
rm stab.out
rm fort.7
rm cosal.out
rm int_n10.out
rm int_n8.out
rm wing.out
rm fort.2
#
#    EXECUTE THE INPUT FILE
#
nice ../stabin.exe<../$case
#
#    EXECUTE THE B.L. CODE
#
nice ../wing.exe< fort.2 > wing.out
#
#    ==================
#    THIS IS THE START OF THE STAB CODE ANALYSIS LOOP
#    ==================
#for run in run1 run25
#
for run in run1 run2 run3 run4 run5 run6 run7 run8 run9 run10
run11 run12 run13 run14 run15 run16 run17 run18 run19 run20
run21 run22 run23
#
#for run in run0 run1 run2 run3 run4 run5 run6 run7 run8 run9
run10 run11 run12 run13 run14 run15 run16 run17 run18
#
do
#    ==================
touch cosal.time
echo "Timing information for running cosal_4.exe:" >>
cosal.time
date >> cosal.time
##  /bin/time nice ../cosal_4.exe < ../$run > cosal.out 2>>
cosal.time
  /bin/time nice ../cosal_4.exe < $run > cosal.out 2>>
cosal.time
date >> cosal.time
#    ==================
#      APPEND THE STAB.OUT INFO, TAKEN FROM THE COSAL.OUT FILE
#    ==================
../getstab.exe < cosal.out >> stab.out
```

```
mv cosal.out cosal.out.bak
done
#   ==================
#     THIS IS THE END OF THE LOOP
#   ==================
nice ../interp_n5_8_p3d.exe<../$case
#
if test "$case" = "fort.73"
then
mkdir stat_c3a1
cp stab.out stat_c3a1/stab_s1.out
cp int_n8.out stat_c3a1/int_n8.s1
cp int_n10.out stat_c3a1/int_n10.s1
fi
#
if test "$case" = "fort.74"
then
mkdir stat_c3a2
cp stab.out stat_c3a2/stab_s2.out
cp int_n8.out stat_c3a2/int_n8.s2
cp int_n10.out stat_c3a2/int_n10.s2
fi
#
if test "$case" = "fort.75"
then
mkdir stat_c3a3
cp stab.out stat_c3a3/stab_s3.out
cp int_n8.out stat_c3a3/int_n8.s3
cp int_n10.out stat_c3a3/int_n10.s3
fi
#
if test "$case" = "fort.76"
then
mkdir stat_c3a4
cp stab.out stat_c3a4/stab_s4.out
cp int_n8.out stat_c3a4/int_n8.s4
cp int_n10.out stat_c3a4/int_n10.s4
fi
#
if test "$case" = "fort.77"
then
mkdir stat_c3a5
cp stab.out stat_c3a5/stab_s5.out
cp int_n8.out stat_c3a5/int_n8.s5
cp int_n10.out stat_c3a5/int_n10.s5
fi
#
if test "$case" = "fort.78"
then
mkdir stat_c3a6
cp stab.out stat_c3a6/stab_s6.out
cp int_n8.out stat_c3a6/int_n8.s6
cp int_n10.out stat_c3a6/int_n10.s6
fi
```


C.2.

```
#
if test "$case" = "fort.79"
then
mkdir stat_c3a7
cp stab.out stat_c3a7/stab_s7.out
cp int_n8.out stat_c3a7/int_n8.s7
cp int_n10.out stat_c3a7/int_n10.s7
fi
#
if test "$case" = "fort.80"
then
mkdir stat_c3a8
cp stab.out stat_c3a8/stab_s8.out
cp int_n8.out stat_c3a8/int_n8.s8
cp int_n10.out stat_c3a8/int_n10.s8
fi
#
if test "$case" = "fort.81"
then
mkdir stat_c3a9
cp stab.out stat_c3a9/stab_s9.out
cp int_n8.out stat_c3a9/int_n8.s9
cp int_n10.out stat_c3a9/int_n10.s9
fi
#
if test "$case" = "fort.82"
then
mkdir stat_c3a10
cp stab.out stat_c3a10/stab_s10.out
cp int_n8.out stat_c3a10/int_n8.s10
cp int_n10.out stat_c3a10/int_n10.s10
fi
#
if test "$case" = "fort.83"
then
mkdir stat_c3a11
cp stab.out stat_c3a11/stab_s11.out
cp int_n8.out stat_c3a11/int_n8.s11
cp int_n10.out stat_c3a11/int_n10.s11
fi
#
if test "$case" = "fort.84"
then
mkdir stat_c3a12
cp stab.out stat_c3a12/stab_s12.out
cp int_n8.out stat_c3a12/int_n8.s12
cp int_n10.out stat_c3a12/int_n10.s12
fi
#
if test "$case" = "fort.85"
then
mkdir stat_c3a13
cp stab.out stat_c3a13/stab_s13.out
cp int_n8.out stat_c3a13/int_n8.s13
```

```
cp int_n10.out stat_c3a13/int_n10.s13
fi
#
if test "$case" = "fort.86"
then
mkdir stat_c3a14
cp stab.out stat_c3a14/stab_s14.out
cp int_n8.out stat_c3a14/int_n8.s14
cp int_n10.out stat_c3a14/int_n10.s14
fi
#
if test "$case" = "fort.87"
then
mkdir stat_c3a15
cp stab.out stat_c3a15/stab_s15.out
cp int_n8.out stat_c3a15/int_n8.s15
cp int_n10.out stat_c3a15/int_n10.s15
fi
#
if test "$case" = "fort.88"
then
mkdir stat_c3a16
cp stab.out stat_c3a16/stab_s16.out
cp int_n8.out stat_c3a16/int_n8.s16
cp int_n10.out stat_c3a16/int_n10.s16
fi
#
if test "$case" = "fort.89"
then
mkdir stat_c3a17
cp stab.out stat_c3a17/stab_s17.out
cp int_n8.out stat_c3a17/int_n8.s17
cp int_n10.out stat_c3a17/int_n10.s17
fi
#
if test "$case" = "fort.90"
then
mkdir stat_c3a18
cp stab.out stat_c3a18/stab_s18.out
cp int_n8.out stat_c3a18/int_n8.s18
cp int_n10.out stat_c3a18/int_n10.s18
fi
#
if test "$case" = "fort.91"
then
mkdir stat_c3a19
cp stab.out stat_c3a19/stab_s19.out
cp int_n8.out stat_c3a19/int_n8.s19
cp int_n10.out stat_c3a19/int_n10.s19
fi
#
if test "$case" = "fort.92"
then
mkdir stat_c3a20
```

```
cp stab.out stat_c3a20/stab_s20.out
cp int_n8.out stat_c3a20/int_n8.s20
cp int_n10.out stat_c3a20/int_n10.s20
fi
#
if test "$case" = "fort.93"
then
mkdir stat_c3a21/
cp stab.out stat_c3a21/stab_s21.out
cp int_n8.out stat_c3a21/int_n8.s21
cp int_n10.out stat_c3a21/int_n10.s21
fi
#
if test "$case" = "fort.94"
then
mkdir test_stat_c3a22/
cp stab.out test_stat_c3a22/stab_s22.out
cp int_n8.out test_stat_c3a22/int_n8.s22
cp int_n10.out test_stat_c3a22/int_n10.s22
fi
#
if test "$case" = "fort.95"
then
mkdir stat_c3a23/
cp stab.out stat_c3a23/stab_s23.out
cp int_n8.out stat_c3a23/int_n8.s23
cp int_n10.out stat_c3a23/int_n10.s23
fi
#
if test "$case" = "fort.96"
then
mkdir stat_c3a24/
cp stab.out stat_c3a24/stab_s24.out
cp int_n8.out stat_c3a24/int_n8.s24
cp int_n10.out stat_c3a24/int_n10.s24
fi
#
done
rm cosal.out
rm int_n10.out
rm int_n8.out
rm stab.out
rm wing.out
rm fort.2
rm fort.7
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1994	Technical Memorandum

4. TITLE AND SUBTITLE

Parametric Study on Laminar Flow for Finite Wings at Supersonic Speeds

5. FUNDING NUMBERS

537-07

6. AUTHOR(S)

Joseph Avila Garcia

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Ames Research Center
Moffett Field, CA 94035-1000

8. PERFORMING ORGANIZATION REPORT NUMBER

A-94146

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM-108852

11. SUPPLEMENTARY NOTES

Point of Contact: Joseph Avila Garcia, Ames Research Center, MS 258-1, Moffett Field, CA 94035-1000; (415) 604-0614

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified — Unlimited
Subject Category 02

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

Laminar flow control has been identified as a key element in the development of the next generation of High Speed Transports. Extending the amount of laminar flow over an aircraft will increase range, payload, and altitude capabilities as well as lower fuel requirements, skin temperature, and therefore the overall cost. A parametric study to predict the extent of laminar flow for finite wings at supersonic speeds was conducted using a computational fluid dynamics (CFD) code coupled with a boundary layer stability code. The parameters investigated in this study were Reynolds number, angle of attack, and sweep. The results showed that an increase in angle of attack for specific Reynolds numbers can actually delay transition. Therefore, higher lift capability, caused by the increased angle of attack, as well as a reduction in viscous drag, due to the delay in transition, can be expected simultaneously. This results in larger payload and range.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Laminar, Supersonic, Wings	111
	16. PRICE CODE: A06

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		