Compressible Vortex Arrays

Thesis by
Kayvan Ardalan

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

1996
(Submitted September 1, 1995)

Acknowledgements

I wish to thank my advisors, Professors Dan Meiron and Dale Pullin, for their invaluable insight, patience and encouragement that lead to the completion of this dissertation. Their support and guidance have made my years as a graduate student at Caltech memorable.

During my stay at Caltech, I have had the pleasure and good fortune to make an abundance of very good friends. They have influenced my life greatly and have helped me grow and mature. I thank you all. Yet, the one constant not only during my graduate studies but throughout my life has been my family. Abolfath, Farideh, Kamran and Sheila's never ending love and support has always been a source of motivation and encouragement for me. Their love is my most prized possesion.

This work has been supported by teaching assistantships in Applied Math, Aeronautics and Applied Physics as well as the Air Force Office of Scientific Research under contract AFOSR F49620-93-1-0338.

Abstract

We construct steady, two dimensional, compressible vortex arrays with specified vorticity distributions. We begin by examining the effects of compressiblity on the structure of a single row of hollow-core, constant pressure vortices. The problem is formulated and solved in the hodograph plane. The transformation from the physical plane to the hodograph plane results in a linear problem that is solved numerically. The numerical solution is checked via a Rayleigh-Janzen expansion. It is observed that for an appropriate choice of the parameters M_∞, the Mach number at infinity, and the speed ratio, a, transonic shock-free flow exists. Also, for a given fixed speed ratio, a, the vortices shrink in size and get closer as the Mach number at infinity, M_∞, is increased. In the limit of an evacuated vortex core, we find that all such solutions exhibit cuspidal behaviour corresponding to the onset of limit lines.

The hollow core vortex array corresponds to a vorticity distribution wherein the vorticity is concentrated on the vortex boundary. In the second part of this thesis, we examine vortex arrays with continuous vorticity distributions. In particular, we construct Stuart-type vortices in a channel by requiring the vorticity distribution to be an exponential function of the stream function of the flow. The problem is formulated and solved in the physical plane. The numerical solution is checked via a Rayleigh-Janzen expansion of the unbounded Stuart vortex solution. It is shown that, in the limit of infinite speed of sound (incompressible flow), as the channel walls tend to infinity, $h \to \infty$, the Stuart vortex solutions are recovered. Furthermore, it is shown that unbounded, compressible Stuart vortices exist and that a generalized Stuart vortex solution is the proper incompressible limit. For a given fixed circulation, Γ, Mach number, M_A, and ϵ, (a measure of the compactness of the vorticity distribution) it is shown that the limit of a very narrow channel, $h \to 0$, is a parallel shear flow. Exact analytical solutions for the compressible parallel shear flow are also found in implicit form.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Many practical problems of interest arising in physics, mathematics and engineering involve the study of vortices and vortex motions. The persistent trailing vortices behind jumbo jets constitute a hazard to other aircraft and this has led to a study of vortex formation by the roll-up of vortex sheets and the decay and interaction of vortex pairs. Attempts to understand two dimensional turbulence have led to studies of the statistical mechanics of random arrays of point vortices. The observations of the turbulent mixing layer (Roshko (1976)) show convincing evidence that the layer consists of a row of quasi two dimensional coherent structures whose amalgamation into larger similar structures produces the growth of the mixing layer. Several theoretical studies attempt to model the formation of these and other structures by studying the interaction of two dimensional vortices. Previous studies, however, have focused exclusively on incompressible flow (Saffman and Sheffield (1977) and Huang and Chow (1982)), and few theoretical studies exist on the interaction of vortices in a compressible medium.

Ringleb (1940), Shapiro (1953), Mack (1960), and Brown (1965) are some of the earlier researchers whose study of the single compressible vortex highlighted the effects of compressibility on the structure of the vortex core as well as its stability characteristics. Moore (1985) studied the effects of compressibility on the speed of a vortex ring. Subsequently, Moore and Pullin (1987) succeeded in constructing the flow field of a translating vortex pair in compressible irrotational flow by transforming the problem to the hodograph plane. In their formulation, each vortex was modelled by a constant pressure stagnant core surrounded by a closed vortex sheet. They realized that for free-boundary problems of this type, this transformation was effective because it fixed the vortex sheet boundary in the hodograph plane and this circumvented the need for curvilinear grids which would be required to solve the problem in the physical plane.

Our aim is to study the basic interactions of vortices in a compressible fluid via a search for steady solutions of the Euler equations corresponding to flows with concentrated vortex cores. An infinite array of point vortices is the simplest model for a shear layer in incompressible flow. It is well known that a hollow core vortex in a compressible medium is the limit of a point vortex in an incompressible medium. Hence, we choose to construct the compressible analogue of the single row vortex array and to study the effects of compressibility on this model. In chapter 2, we shall extend the results of Baker, Saffman and Sheffield (1976) to include compressibility. This model of a shear layer has the vorticity in the flow concentrated on a vortex sheet placed on the vortex boundary. This specific choice of vorticity distribution makes the vortex boundary a streamline of the flow. As will be shown in subsequent chapters, transforming this free boundary problem to the hodograph plane results in a linear problem that is solved numerically. It is shown in section 2.5 that use of the Sherman-Morrison formula [9] allows for the application of a discrete Fourier transform to the discretized problem. This reduces the storage requirements for the numerical algorithm which in turn allows for increased resolution. A parameter search of the problem shows the regions of the flow that allow for transonic shock-free flow. As a result of the increased resolution, it's shown that no evacuated vortex cores exist without the appearance of a limit line. Furthermore, it is shown that the vortices shrink in size and get closer together as the Mach number in the flow is increased.

In chapter 3, we shall extend the result of Stuart (1967) which is an exact solution of the nonlinear, inviscid equations of motion. The distribution of vorticity throughout the flow field is continuous and is given by $\omega = \exp(-2\psi)$. The stream function of the flow is given by

$$\psi = \log(\kappa \cosh 2\pi y/\lambda + \sqrt{\kappa^2 - 1} \cos 2\pi x/\lambda), \tag{1.1}$$

where $1 \leq \kappa \leq \infty$ is a parameter of the flow. The limit of $\kappa = 1$ corresponds to a parallel shear flow and the limit as $\kappa \to \infty$ is an infinite row of point vortices separated a fixed distance λ apart. The stability of this solution has been studied numerically (Pierrehumbert and Widnall (1982)) and it has been shown that Stuart vortices display some of the properties observed by studying mixing layers experimentally (Roshko (1976)). Specifically, it is shown that shear-layer growth by vortex amalgamation and generation of three dimensionality by localized pairing are associated with the subharmonic instabilities. However, Drazin and Reid (1981) remark that the Stuart vortex solution, although intriguing, covers special cases and is not yet related to the main body of the theory of nonlinear stability. While showing that Stuart vortices can exist in the Garcia model of a stratified shear layer, Mallier (1994) suggests that the Stuart vortex solution is a special case of a much larger family of vortices. We suggest an alternate derivation and call the final result the generalized Stuart vortex solutions since it provides more insight on how to continue the Stuart vortices to compressible flow. It must be noted that the generalized Stuart vortices can be transformed to the Stuart vortex solution through an appropriate translation and stretching of the coordinate axes. Furthermore, we construct Stuart-type vortices with walls in a compressible medium, and by increasing the channel width, we show numerically that steady, two dimensional Stuart vortices in an unbounded compressible fluid exist. The numerical scheme uses a Newton iteration to solve the nonlinear governing equations which are derived in chapter 3. It is shown that decreasing the wall height tends to eliminate the recirculation regions present in Stuart vortex solutions. We also find exact analytic solutions for compressible, parallel flow shear layers that could be of use in stability studies.

In both of these investigations, we validate the numerical solutions via a Rayleigh-Janzen expansion. In the case of the hollow core vortex array of chapter 2, we were able to find an analytic expression for the first order compressibility correction, whereas in chapter 3 it was necessary to solve for it numerically.

Chapter 2

The hollow core vortex array

2.1 The physical plane

We consider an infinite linear array of identical vortices lying on the x-axis each separated by a distance L measured from the vortex center (cf. Figure 2.1). Each vortex is hollow (i.e., has a stagnant, constant pressure core). Since the flow is steady and the pressure is constant inside the cores, the fluid speed on the boundary of the vortex must be a constant value, q_v. Outside the cores, the flow is that of an ideal gas and it is assumed to be irrotational, compressible and homentropic. The vortices have the same circulation, Γ, such that at large distances the flow becomes asymptotic to that produced by a vortex sheet lying along the x-axis.

We shall seek steady solutions of the configuration sketched in Figure 2.1 in which the vortices have fore-and-aft and top and bottom symmetry, i.e., each vortex is symmetrical about the x-axis and a line parallel to the y-axis through the vortex center. Hence, it will be sufficient to investigate the details of the flow in the geometry depicted in Figure 2.2. The equation

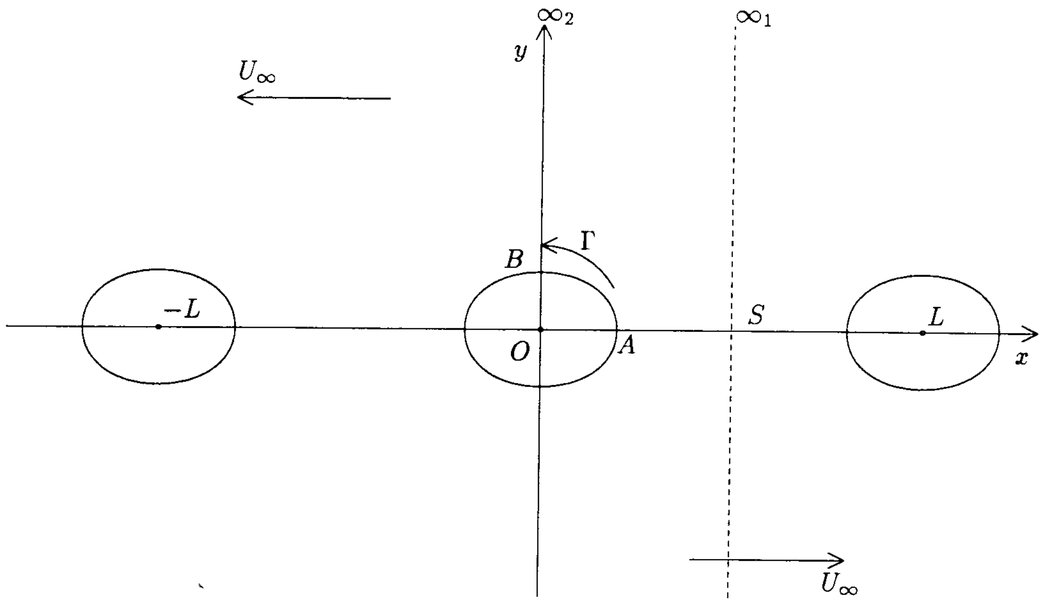


Figure 2.1: A sketch of the problem in the physical space. The closed curves show the vortex boundaries on which the speed is q_v.

of continuity in the case of steady two dimensional compressible flow can be written as,

$$\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} = 0, \tag{2.1}$$

where ρ is the density and u and v are the horizontal and vertical components of the velocity vector. We can satisfy this identically by introducing a stream function ψ defined so that

$$\rho u = -\rho_o \frac{\partial \psi}{\partial y},$$

$$\rho v = \rho_o \frac{\partial \psi}{\partial x},$$

where ρ_o is any reference density. We shall identify the density ρ_o with the density of the flow at infinity (ρ_∞) and set $\rho_\infty = 1$ without loss of generality.

The governing equation in this problem is given by Bernoulli's theorem. For homentropic flow of an ideal gas, if c is the speed of sound and p is the pressure, we have $c^2 = \gamma p/\rho$ and therefore conservation of energy requires

$$\frac{c^2}{\gamma - 1} + \frac{1}{2}q^2 = \frac{c_s^2}{\gamma - 1}, \tag{2.2}$$

where $q = \sqrt{u^2 + v^2}$ is the speed of the flow and c_s is the speed of sound at the stagnation condition (i.e., when $q = 0$).

We seek the solution of these equations in the computational domain depicted in Figure 2.2, which shows a sketch of the streamlines. The point S corresponds to the stagnation point where the speed of the fluid q is zero. The curve connecting points A and B represents the boundary of the vortex. The vertical lines from B and S are assumed to extend to infinity where the velocity of the flow has a single speed, q, and phase direction, θ, namely, $q = q_\infty$ and $\theta = \pi$. On the two vertical boundaries ∞_1-S and B-∞_2, even though the speed of the flow varies, the assumed symmetries determine the phase angle of the velocity vector to be $\theta = \pi$. On the symmetry line $S - A$, the flow angle is $\theta = \pi/2$. Even though it is possible to proceed with the solution of this problem in the physical plane, we instead transform to the hodograph plane where the geometry of the flow simplifies to a rectangle and the governing equation is Chaplygin's equation, which is a linear equation for the stream function.

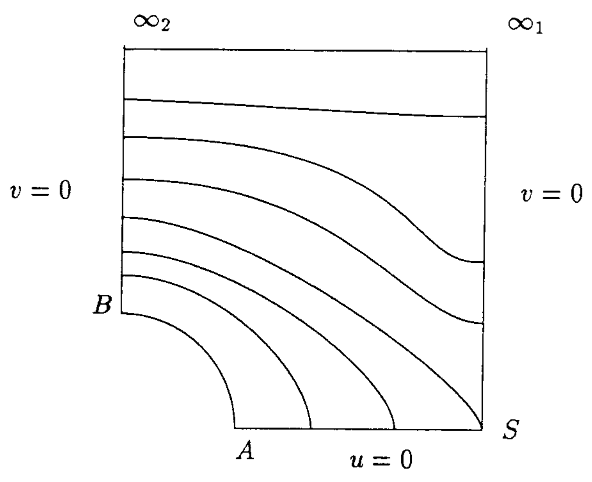


Figure 2.2: A description of the computational domain in physical space.

2.2 The hodograph plane

2.2.1 Chaplygin's equation

The velocity potential of an irrotational compressible flow satisfies a non-linear partial differential equation. When (q, θ) are taken as variables, the equation becomes linear [27]. This leads to a quasi-conformal map from the physical plane ((x,y) space) to the hodograph plane ((q,θ) space) which we shall summarize for completeness. Along with the stream function ψ defined earlier, we can introduce the velocity potential ϕ and write

$$\begin{bmatrix} -d\phi \\ (-\rho_o/\rho) \quad d\psi \end{bmatrix} = \begin{bmatrix} u & v \\ -v & u \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix},$$

and using methods of complex variables, it is easily verified [16] that

$$dz = -\frac{e^{\imath\theta}}{q}(d\phi + \imath\frac{\rho_o}{\rho}d\psi), \tag{2.3}$$

where $z = x + \imath y$ and $u - \imath v = qe^{\imath\theta}$. It follows that

$$\frac{\partial z}{\partial q} = -\frac{e^{\imath\theta}}{q}(\frac{\partial\phi}{\partial q} + \imath\frac{\rho_o}{\rho}\frac{\partial\psi}{\partial q}), \tag{2.4}$$

$$\frac{\partial z}{\partial\theta} = -\frac{e^{\imath\theta}}{q}(\frac{\partial\phi}{\partial\theta} + \imath\frac{\rho_o}{\rho}\frac{\partial\psi}{\partial\theta}). \tag{2.5}$$

Cross differentiation of the above equations then leads to a compatibility condition akin to the Cauchy-Riemann equations:

$$\frac{\partial\phi}{\partial q} = q\frac{\partial\psi}{\partial\theta}\frac{\partial}{\partial q}(\frac{1}{\rho q}), \tag{2.6}$$

and

$$\frac{\partial\phi}{\partial\theta} = \frac{q}{\rho}\frac{\partial\psi}{\partial q}. \tag{2.7}$$

These are the equations of the hodograph plane. Eliminating ϕ from equations (2.4) and (2.5) and noting that, in the hodograph plane, the density is a function of the speed only, $\rho = \rho(q)$, leads to Chaplygin's equation,

$$q^2\left[1 - \frac{q^2(\gamma-1)}{2c_s^2}\right]\frac{\partial^2\psi}{\partial q^2} + q\left[1 - \frac{q^2(\gamma-3)}{2c_s^2}\right]\frac{\partial\psi}{\partial q} + \left[1 - \frac{q^2(\gamma+1)}{2c_s^2}\right]\frac{\partial^2\psi}{\partial\theta^2} = 0. \tag{2.8}$$

2.2.2 Scaling of variables

We next consider the image of our problem in the hodograph plane. Before doing so we scale all speeds by the speed on the vortex boundary, q_v. Furthermore, we shall scale the stream function by the strength of the singularity at infinity in the hodograph plane, which we denote by $\mathcal{A}$. The singularity at infinity will be discussed in detail in section 2.2.4. This along with our earlier choice of the reference density, namely $\rho_\infty = 1$, leads to the following relationship between the length, time and mass scales L', T', M'.

$$L'^3 \rho_\infty M'^{-1} = 1, \quad q_v T' L'^{-1} = 1, \quad \mathcal{A} L'^{-2} T' = 1.$$

At this point we have completed the specification of scales for our problem. It is important to note here that the distance between the vortices L has not been used for scaling and that it shall be determined as part of the solution.

The use of equation (2.2) allows us to express the speed of sound at the stagnation point as

$$\frac{1}{c_s^2} = \frac{2M_\infty^2}{2+(\gamma-1)M_\infty^2} \frac{1}{a^2 q_v^2} \tag{2.9}$$

where $a = q_\infty / q_v$ and the Mach number at infinity, $M_\infty = q_\infty / c_\infty$. We shall define $\beta = q_v^2/2c_s^2$ and hence can rewrite Chaplygin's equation as follows:

$$q^2\left[1-\beta(\gamma-1)q^2\right]\psi_{qq} + q\left[1-\beta(\gamma-3)q^2\right]\psi_q + \left[1-\beta(\gamma+1)q^2\right]\psi_{\theta\theta} = 0. \tag{2.10}$$

2.2.3 The flow in the hodograph plane

To determine the image of the flow in the hodograph plane, we must return to Figure 2.2. Since at the stagnation point S the speed is zero, the streamlines form a saddle point pattern at which the flow angle takes on all values between $\pi/2$ and π. Therefore, the stagnation point maps into the vertical line $q = 0$ in Figure 2.3. Moving on to the boundary of the vortex, we know that the speed is constant ($q = q_v$), so after scaling by q_v, the vertical line $q = 1$ is the image of the vortex boundary in the hodograph plane. The streamline at infinity corresponds to a single speed, $q = q_\infty$, and direction, $\theta = \pi$, and so we see that an entire streamline is transformed into a point singularity in the hodograph plane. We defer a discussion of the nature of this singular tranformation for the moment and discuss boundary conditions.

Since the stream function is arbitrary to within a constant, we will choose $\psi = 0$ on the boundary of the vortex without loss of generality. On the boundary $A - S$, we have $u = 0$. Additionally, Figure 2.2 shows that this boundary is horizontal, $dy = 0$. Using this information along with equations (2.1), (2.4) and (2.5), we obtain that $\partial\psi/\partial\theta = 0$ on $\theta = \pi/2$. On the segment from $B - \infty$ and $\infty - S$, we have $v = 0$ and the streamlines have a single direction $\theta = \pi$. It is also clear that here the boundary is vertical, $dx = 0$. The use of the hodograph transformation then leads us to the condition that $\partial\psi/\partial\theta = 0$ on $\theta = \pi$. Having determined these boundary conditions, we can now investigate the nature of the solution near the stagnation point. An asymptotic analysis of Chaplygin's equation near $q = 0$ shows that to leading order the equation reduces to

$$q^2\psi_{qq} + q\psi_q + \psi_{\theta\theta} = 0,$$

which along with the boundary conditions

$$\psi_\theta = 0 \qquad\qquad \theta = \pi/2,\ \pi$$

has the series solution,

$$\psi \sim c_1 + c_2 q^2 cos(2\theta) + O(q^4)$$

where c_1 and c_2 are constants. Hence, setting $\partial\psi/\partial q = 0$ on $q = 0$ is consistent. The problem to be solved in the hodograph plane is depicted in Figure 2.3.

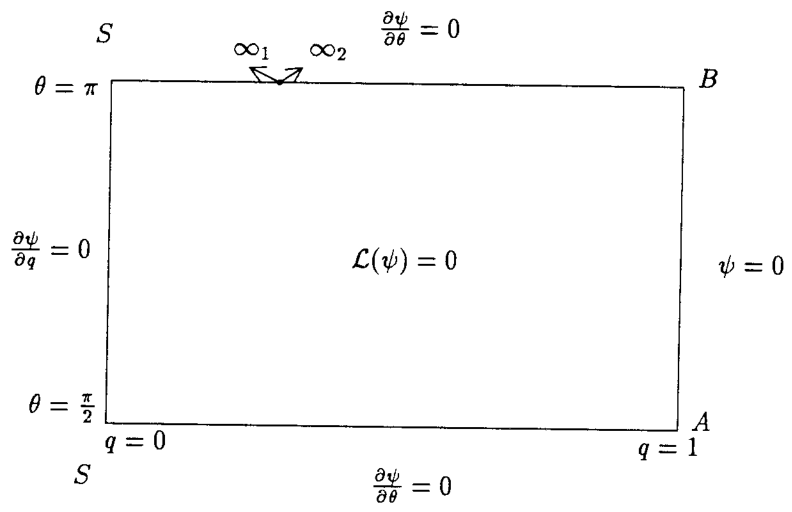

S
∞_1
∞_2
$\frac{\partial\psi}{\partial\theta} = 0$
$\theta = \pi$
B
$\frac{\partial\psi}{\partial q} = 0$
$\mathcal{L}(\psi) = 0$
$\psi = 0$
$\theta = \frac{\pi}{2}$
A
$q = 0$
$q = 1$
S
$\frac{\partial\psi}{\partial\theta} = 0$

Figure 2.3: Computational Domain in Hodograph Plane.

2.2.4 The singularity at infinity

Using small-disturbance theory it is possible, by means of a simple transformation, to reduce all subsonic flow problems to an equivalent incompressible flow problem. This is known as the Prandtl- Glauert rule [5]. The incompressible velocity potential is given by $\Phi(x, y)$ and the equivalent subsonic potential is given by $\Phi'(x, \eta)$ where

$$\Phi = \frac{1}{\sqrt{1 - M_\infty^2}} \Phi',$$

$$\eta = \sqrt{1 - M_\infty^2} y.$$

We assume that the flow at infinity is subsonic since otherwise a shock-free flow field seems unlikely [18]. Assuming small disturbances at infinity and distorting the incompressible flow solution for a row of equal point vortices [12] as suggested by Prandtl-Glauert theory, we find that the velocity potential is

$$\phi = \frac{\Gamma}{2\pi} \arctan[\cot(\pi x/L) \tanh(\pi \sqrt{1 - M_\infty^2} y/L)]. \tag{2.11}$$

To leading order, therefore, the stream function is given by

$$\psi = \frac{\Gamma}{2L} y \tag{2.12}$$

and the corresponding velocity field is

$$u = \frac{\Gamma}{2L} \frac{\sinh(2\pi \sqrt{1 - M_\infty^2} y/L)}{\cosh(2\pi \sqrt{1 - M_\infty^2} y/L) - \cos(2\pi x/L)} \tag{2.13}$$

and

$$v = \frac{\Gamma \sqrt{1 - M_\infty^2}}{2L} \frac{\sin(2\pi x/L)}{\cosh(2\pi \sqrt{1 - M_\infty^2} y/L) - \cos(2\pi x/L)}. \tag{2.14}$$

In order to find the nature of the singularity in the hodograph plane, we need to eliminate x and y from (2.12), (2.13) and (2.14). Using local quasi-polar coordinates (s, δ) centered at the point at infinity in the hodograph plane defined by

$$se^{\imath\delta} = (q - a) + \imath\frac{\theta - \pi}{\sqrt{1 - M_\infty^2}}, \tag{2.15}$$

we can show (Appendix A) that, for small s, the leading order stream function is given by

$$\psi = \mathcal{A}\log(s). \tag{2.16}$$

This form of the singularity indicates that as $s \to 0$, regardless of direction δ, all higher order corrections to the logarithm go to zero. We have now posed the problem in the hodograph plane. Its solution $\psi(q, \theta)$ is a two parameter family depending on $a = q_\infty/q_v$, the location of the singularity in the hodograph plane and $M_\infty = q_\infty/c_\infty$, the Mach number at infinity. Note that as discussed in section 2.2.2, the strength of the vortex $(\mathcal{A} = 1)$ is scaled out of the problem.

2.3 The incompressible problem

We begin by obtaining an analytical solution in the hodograph plane for the $M_\infty = 0$ limit of the single row. Setting $M_\infty = 0$ reduces (2.10) to Laplace's equation, so by using equation (2.16) we deduce that the correct form of the singular solution near the point at infinity is given by a point vortex positioned at the singular point $(q = a, \theta = \pi)$. Using the law of cosines, we find that this is given by

$$\psi_s = \log(q^2 + a^2 + 2aq\cos\theta). \tag{2.17}$$

Now all that is necessary to complete the solution is to satisfy the boundary conditions. We apply the method of images to obtain the solution

$$\psi_o = \log\left[\frac{(q^2+a^2+2aq\cos\theta)(q^2+a^2-2aq\cos\theta)}{(q^2a^2+1+2aq\cos\theta)(q^2a^2+1-2aq\cos\theta)}\right]. \tag{2.18}$$

This solution agrees with that given by [2] who solved the same problem using the ideas of free-streamline theory. For comparison, we find a parametric representation for the vortex boundary centered at the origin. The nondimensional coordinates of the boundary are given by

$$x = -2\frac{a^2+1}{a}\left[\tan^{-1}\left[\frac{(a+1)\sin\theta}{(\cos\theta+1)(a-1)}\right] - \tan^{-1}\left[\frac{(a-1)\sin\theta}{(\cos\theta+1)(a+1)}\right]\right] \tag{2.19}$$

$$y = \frac{a^2-1}{a}\left[\log(a^2+1+2a\cos\theta) - \log(a^2+1-2a\cos\theta)\right]. \tag{2.20}$$

As shown in Figure 2.4, Baker, Saffman and Sheffield (1976) parameterize their solutions by the shape ratio $R = P/2L$ where P is the perimeter of the vortex and L is the separation between the vortices. The circulation Γ about each vortex is then related to q_v, the speed of the vortex boundary, by

$$\Gamma = Pq_v.$$

Furthermore, at large distances, the array looks like a vortex sheet of strength $2q_\infty$, where

$$q_\infty = \frac{1}{2}\Gamma/L.$$

Hence, conservation of circulation then requires that $a = R = q_\infty/q_v$, the location of the singularity in the hodograph plane. Note that the shape ratio is restricted to be in the range $0 < R < 1$. The limit $R = 0$ corresponds to an array of point vortices. The opposite limit $R = 1$ corresponds to a vortex sheet in which each vortex is pulled out longitudinally and squeezed sideways to lie along a length L of the x-axis. The significance of this solution for our problem is that we have obtained the zeroth order solution of a perturbation solution (the Rayleigh-Janzen expansion) with M_∞^2 as the small parameter. An examination of the terms of the operator $\mathcal{L}(\psi)$ indicates that compressibility is introduced as regular corrections to the incompressible flow indicating the possible existence of a regular perturbation expansion in powers of M_∞. Below we construct several terms of this solution.

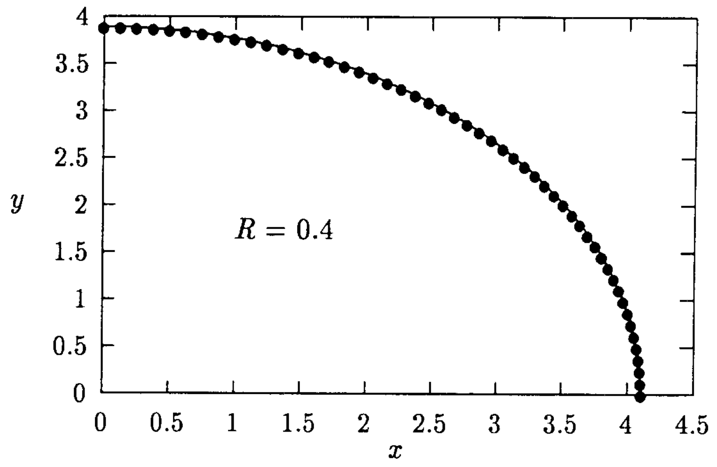


Figure 2.4: Shape of vortex boundary for $M_\infty = 0$, $a = 0.4$. Solid circles represent Baker, Saffman and Sheffield (1976) solution. Solid line is a plot of equations (2.19) and (2.20).

2.4 Perturbation solution

We construct an approximate series solution to the problem depicted in Figure 2.3 valid when $M_\infty \ll 1$. The problem is formulated as

$$q^2(1-\beta(\gamma-1)q^2)\psi_{qq}+q(1-\beta(\gamma-3)q^2)\psi_q+(1-\beta(\gamma+1)q^2)\psi_{\theta\theta}=0, \quad (2.21)$$

with the boundary conditions given by

$$\begin{aligned} \frac{\partial\psi}{\partial q} &= 0 \quad &\text{at} \quad & q=0 \\ \psi(q,\theta) &= 0 \quad &\text{at} \quad & q=1 \\ \frac{\partial\psi}{\partial\theta} &= 0 \quad &\text{at} \quad & \theta=\frac{\pi}{2} \text{ and } \pi, \end{aligned} \quad (2.22)$$

where $\beta = \frac{M_\infty^2/a^2}{2+(\gamma-1)M_\infty^2}$. The solution we seek has the Rayleigh-Janzen form,

$$\psi(q,\theta) = \psi_o + M_\infty^2\psi_1 + O(M_\infty^4). \quad (2.23)$$

As noted above at zeroth order, we have the incompressible solution which is given by equation (2.18). The second order problem is given by

$$q^2\psi_{1_{qq}} + q\psi_{1_q} + \psi_{1_{\theta\theta}} = \frac{q^2}{a^2}(\psi_{o_{\theta\theta}} - q\psi_{o_q}) \quad (2.24)$$

with the same boundary conditions as mentioned earlier for equation (2.21). Note that this system has many singular homogenous solutions. According to Prandtl-Glauert theory, we must ensure that the incompressible solution is the dominant singularity as the point at infinity is approached. The only regular homogeneous solution to this problem is the trivial solution $\psi(q,\theta)=0$. Hence, finding a particular solution will then lead to a unique solution to this order. By inspection, a particular solution is given by

$$\psi_{1_p} = \frac{q^3}{4}\frac{\partial \psi_o}{\partial q},$$

but closer inspection of the above result shows that this is more singular than the incompressible solution. Note that equation (2.16) shows that $\psi \sim \log s$ (incompressible solution) and hence the particular solution is more singular since $\psi_{1_p} \sim 1/s$. Therefore, we find a complimentary function to add to this particular solution such that the final result is regular at the point at infinity and that all boundary conditions are satisfied. This leads to a solution valid to $O(M_\infty^4)$:

$$\psi(q,\theta) = \psi_o + M_\infty^2 \psi_1 \tag{2.25}$$

where ψ_o is given by equation (2.18) and the compressibility correction is given by

$$\begin{aligned} \psi_1 \;=\; & -q^2 \frac{(a^4-1)(1-q^2/a^2)}{(q^2+a^2+2aq\cos\theta)(q^2+a^2-2aq\cos\theta)} f_1(q,\theta) \\ & -2q^2(1-a^2) f_2(q,\theta) \\ & + \left[\frac{1-a^2}{a^2}\right] \end{aligned} \tag{2.26}$$

where

$$f_1(q,\theta) = \frac{a^4q^2 - 2a^2q^4\cos\theta^2 + q^4a^2 + a^2 - 2a^2\cos\theta^2 + q^2}{(q^2a^2+1+2aq\cos\theta)(q^2a^2+1-2aq\cos\theta)} \tag{2.27}$$

and

$$f_2(q,\theta) = \frac{q^2a^2+1-2\cos\theta^2}{(q^2a^2+1+2aq\cos\theta)(q^2a^2+1-2aq\cos\theta)}. \tag{2.28}$$

It is essential to verify that the solution above maintains the integrity of our original mapping. We check this by verifying that the total distance traversed around a closed loop in the entire hodograph plane (including the boundaries) yields zero so that the mapping remains one to one. Equivalently, referring to Figure 2.2, we can show that the horizontal distance traversed from S to B, x_{BS}, is equal to that traversed going from ∞_1 to ∞_2, x_∞. Equation (2.3) gives

$$dx = \left[\cos\theta \frac{\partial}{\partial q}(\frac{1}{\rho q})\psi_\theta - \frac{\sin\theta}{\rho q}\psi_q\right] dq + \left[\frac{\cos\theta}{\rho}\psi_q - \frac{\sin\theta}{\rho q}\psi_\theta\right] d\theta$$

for measuring horizontal distances in the hodograph plane. Hence, the distance from the center of the vortex to the stagnation point, x_{BS}, is given by

$$x_{BS} = \int_{\pi/2}^{\pi} \left(\frac{\cos\theta}{\rho}\psi_q - \frac{\sin\theta}{\rho q}\psi_\theta\right)_{q=1} d\theta + \int_0^1 \left(-\frac{\sin\theta}{\rho q}\psi_q\right)_{\theta=\pi/2} dq.$$

This distance must be equal to the length of the streamline at infinity which is given by

$$x_\infty = \lim_{\hat{s}\to 0} \int_0^{-\pi} dx(\hat{s},\hat{\delta})\, d\hat{\delta},$$

where $\hat{s}$ and $\hat{\delta}$ are local polar coordinates centered about the singularity in the hodograph plane. In order to check closure, we must note that the strength of the vortex is an unknown function of the Mach number so that the appropriate expansion for the stream function is given by

$$\psi(q,\theta) = (1 + \xi_1 M_\infty^2 + O(M_\infty^4))\,\psi_0(q,\theta) + M_\infty^2\psi_1(q,\theta) + O(M_\infty^4).$$

Since the density is only a function of the speed,

$$\rho = \left[1 + \frac{\gamma - 1}{2} M_\infty^2 (1 - \frac{q^2}{a^2})\right]^{\gamma-1},$$

the appropriate expansion in Mach numbers is given by

$$\rho = 1 + \rho_1(q)\, M_\infty^2 + O(M_\infty^4),$$

where $\rho_1 = \frac{1}{2}(1 - \frac{q^2}{a^2})$. Substituting the above results into the expression for x_{BS} and expanding in the Mach number, we find that

$$x_{BS} = x_{BS}^0 + M_\infty^2 x_{BS}^1 + O(M_\infty^4),$$

where

$$x_{BS}^0 = \int_{\pi/2}^{\pi} \left(\cos\theta\, \psi_{0_q} - \frac{\sin\theta}{q} \psi_{0_\theta}\right)_{q=1} d\theta + \int_0^1 \left(-\frac{\sin\theta}{q} \psi_{0_q}\right)_{\theta=\pi/2} dq$$

and that

$$x_{BS}^1 = \eta_1 + \eta_2 + 2\pi\xi_1/a$$

where

$$\eta_1 = \int_{\pi/2}^{\pi} \left(\cos\theta\, \psi_{1_q} - \frac{\sin\theta}{q} \psi_{1_\theta}\right)_{q=1} d\theta + \int_0^1 \left(-\frac{\sin\theta}{q} \psi_{1_q}\right)_{\theta=\pi/2} dq,$$

$$\eta_2 = \int_{\pi/2}^{\pi} \left[-\rho_1(\cos\theta\, \psi_{0_q} - \frac{\sin\theta}{q} \psi_{0_\theta})\right]_{q=1} d\theta + \int_0^1 \left(\rho_1 \frac{\sin\theta}{q} \psi_{0_q}\right)_{\theta=\pi/2} dq.$$

Evaluating the above integrals, we have that

$$x_{BS} = 2\pi/a + (\frac{2\pi\xi_1}{a} + \pi/a)\, M_\infty^2 + O(M_\infty^4).$$

In finding the value of x_∞, we define a local coordinate system centered at the singular point $(q = a, \theta = \pi)$,

$$q = \hat{s}\cos\hat{\delta} + a,$$

$$\theta = \hat{s}\sin\hat{\delta} + \pi.$$

This makes the expression for $dx = dx(\hat{s}, \hat{\delta})$ a function of the local coordinates, hence evaluating the integrals gives

$$x_\infty = 2\pi/a + (\frac{2\pi\xi_1}{a} + \pi/a)\, M_\infty^2 + O(M_\infty^4).$$

Hence, we see that our perturbation solution, equation (2.25), satisfies the closure condition to $O(M_\infty^4)$ automatically since $x_{BS} = x_\infty$. With this solution valid for small M_∞, we proceed to discuss the solution at larger values of M_∞ which is obtained numerically.

2.5 Numerical method

Since the nature of the singularity is known from our earlier analysis, we opted to solve the following modified but equivalent problem:

$$\begin{aligned} \psi &= \psi_s + \psi_r \\ \mathcal{L}(\psi_r) &= -\mathcal{L}(\psi_s) \end{aligned} \qquad (2.29)$$

where

$$\begin{aligned}
\psi_s &= \log\left[(q-a)^2 + a^2\frac{(\theta-\pi)^2}{1-M_\infty^2}\right] \\
&- \frac{a^2(\theta-\pi)^2}{(1-M_\infty^2)(q-a)^2 + (\pi^2a^2/4)}.
\end{aligned} \tag{2.30}$$

This particular singular form of ψ_s is chosen for two reasons. The main advantage is that a local asymptotic analysis near the point at infinity shows that all other higher order terms vanish in the limit as $(q,\theta) \to (a,\pi)$ as shown in Appendix A. This ensures ψ_r is finite at (a,π). In addition, the top and bottom boundary conditions remain unchanged from those for ψ. The remaining boundary conditions for the modified problem are

$$\begin{aligned}
\frac{\partial\psi_r}{\partial q} &= -\frac{\partial\psi_s}{\partial q}, \quad q=0 \\
\psi_r(q,\theta) &= -\psi_s(q,\theta), \quad q=1 \\
\frac{\partial\psi_r}{\partial\theta} &= 0, \qquad \theta=\frac{\pi}{2}, \ \ \pi.
\end{aligned} \tag{2.31}$$

The modified problem described above was solved numerically using second-order centered differences on a fixed grid

$$\begin{aligned}
q_i &= (i-1)\Delta q, & i &= 1,...,N \\
\theta_j &= \pi/2 + (j-1)\Delta\theta, & j &= 1,...,M+1
\end{aligned} \tag{2.32}$$

where $\Delta q = 1/(N-1)$ and $\Delta\theta = \pi/(2M)$, and we define $\psi_{i,j} \equiv \psi_r(q_i,\theta_j)$. The finite difference form of equation (2.29) is given by

$$\left[\frac{A_{i,j}}{\Delta q^2} + \frac{B_{i,j}}{2\Delta q}\right]\psi_{i+1,j} \ - \ 2\left[\frac{A_{i,j}}{\Delta q^2} + \frac{C_{i,j}}{\Delta\theta^2}\right]\psi_{i,j} + \left[\frac{A_{i,j}}{\Delta q^2} - \frac{B_{i,j}}{2\Delta q}\right]\psi_{i-1,j}$$

$$+ \quad \frac{C_{i,j}}{\Delta\theta^2}\left(\psi_{i,j+1} + \psi_{i,j-1}\right) = f_{i,j} \tag{2.33}$$

where $f_{i,j} = -\mathcal{L}(\psi_s(q_i, \theta_j))$ and the coefficients are defined as follows:

$$\begin{aligned} A_{i,j} &= q_i^2\left[1 - \beta(\gamma - 1)q_i^2\right], \\ B_{i,j} &= q\left[1 - \beta(\gamma - 3)q_i^2\right], \\ C_{i,j} &= \left[1 - \beta(\gamma + 1)q_i^2\right]. \end{aligned} \tag{2.34}$$

At the top and bottom boundaries, where $\theta = \pi$ and $\theta = \frac{\pi}{2}$, the derivative conditions are satisfied by adding fictitious points parallel to these boundaries and applying centered differences except for the point at $(q_i, \theta_j) = (a, \pi)$. We can not add fictitious points there because the forcing term in equation (2.29) is not defined. Hence, since we are solving for ψ_r, which is the regular part of our solution, we found it sufficient to impose the derivative condition using forward differencing:

$$\begin{aligned} \hat{\imath} &= \frac{a}{\Delta q} + 1, \\ -3\psi_{\hat{\imath},M+1} + 4\psi_{\hat{\imath},M} - \psi_{\hat{\imath},M-1} &= 0, \end{aligned} \tag{2.35}$$

where $\hat{\imath}$ is the location of the singularity on the grid.

The use of forward differencing at $q = 0$ also proved to be the most effective means for avoiding unphysical θ-dependence of the stagnation value. Implementation of the right-hand side Dirichlet condition ($\psi_r(1, \theta) = -\psi_s(1, \theta)$) is straightforward. Finally, the problem is reduced to one of linear algebra and solving a system $\mathbf{A} \cdot \mathbf{x} = \mathbf{B}$. At this stage, the main difficulty is resolution since the size of $\mathbf{A}$ is $(N \cdot (M + 1) \times N \cdot (M + 1))$. Figure 2.5 shows

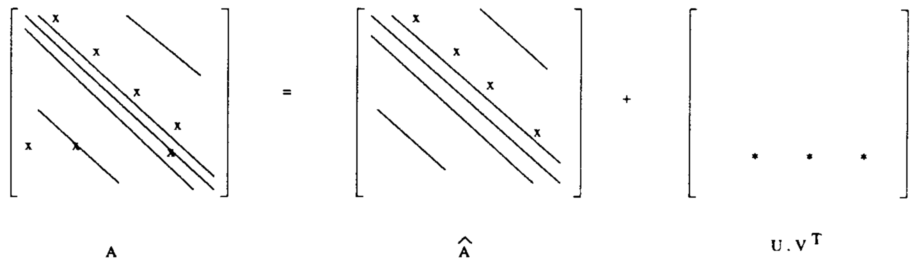


Figure 2.5: Decompostion of finite difference coefficient matrix. The cross and asterisks symbols emphasize the location of the corrected rows. Matrix size is $(N \cdot (M+1) \times N \cdot (M+1))$.

a schematic of the banded coefficient matrix **A** and its decomposition. If not for the form of the θ-derivative at the singular point, **A** would have been amenable to transform methods since it would have been a block tridiagonal matrix; hence, we chose to use the capacitance matrix approach to change **A** appropriately. The decomposition of matrix **A** can be written as

$$\mathbf{A} = \hat{\mathbf{A}} + \mathbf{U} \cdot \mathbf{V}^T.$$

The Woodbury formula, which is the block-matrix version of the Sherman-Morrison formula [9], relates $\mathbf{A}^{-1}$ to its decompostion as

$$(\hat{\mathbf{A}} + \mathbf{U} \cdot \mathbf{V}^T)^{-1} = \hat{\mathbf{A}}^{-1} - [\hat{\mathbf{A}}^{-1} \cdot \mathbf{U} \cdot (1 + \mathbf{V}^T \cdot \hat{\mathbf{A}}^{-1} \cdot \mathbf{U})^{-1} \cdot \mathbf{V}^T \cdot \hat{\mathbf{A}}^{-1}], \quad (2.36)$$

where the term $(1 + \mathbf{V}^T \cdot \hat{\mathbf{A}}^{-1} \cdot \mathbf{U})^{-1}$ is known as the capacitance matrix and it has dimensions (pxp) where p is the number of corrected rows. Since we have Neumann boundary conditions in the θ-direction, we apply a discrete cosine transform [19] on $\hat{\mathbf{A}}$. Since we use forward difference formulas at $q = 0$, we choose to apply the Woodbury formula one more time so that we have tridiagonal matrices. Making this extra effort increases our resolution compared to inverting the original banded matrix since we have minimized the storage requirements for $\mathbf{A}$.

Note that to obtain ψ_r we solve a linear system. This was not obvious at the outset. Indeed Moore and Pullin (1987) find residual nonlinearity in their formulation of the compressible vortex pair problem in the form of a forced closure of the physical plane when reconstructed from the solution of the hodograph problem. This led us to expect that an iterative solution procedure resulting from the non-linear nature of compressible flow problems would be required to solve this problem. However, the transformation to the hodograph plane in this specific example has lead to an entirely linear problem. Our final solution is written as

$$\psi(q,\theta) = \psi_s(q,\theta) + \psi_r(q,\theta).$$

For $M_\infty \ll 1$, we compare our numerical solution with our perturbation solution for various values of q and θ and successfully verify that the error is $O(M_\infty)^4$. Table (2.1) shows the results from one of such tests.

2.6 Results

We perform a parameter search to determine those solutions that are physically relevant. It is well known (Landau & Lifschitz 1959) that if the Jacobian of the hodograph transformation vanishes at any point, the solution will exhibit cuspidal behaviour corresponding to the onset of limit lines [11].

q	$\psi(q,\theta=\pi/2)$	
	Numerical	Perturbation
0.00	-3.6129	-3.6127
0.20	-3.1786	-3.1784
0.40	-2.2798	-2.2796
0.80	-0.6199	-0.6197
1.00	0.0000	0.0000

Table 2.1: A comparison of the numerical and perturbation results at $M_\infty = 0.1$ and $a = 0.4$.

Hence, in our search of the parameter space $(a,\ M_\infty)$, we monitored the Jacobian for changes of sign. The search was done at a fixed value of a, stepping in the Mach number in increments of 0.001, on a $(N, M+1) = (500 \times 128)$ grid, until a limit line was approached. Note that the computation was stopped at the first sight of a limit line and then the resolution was increased to $(N, M+1) = (1000 \times 256)$ and (2000×256) to ensure that the results are grid independent. The level of accuracy of the numerical algorithm for these computations was up to six digits. The outcome of this investigation is depicted in Figure 2.6 which summarizes the range of solutions found. It is evident from the figure that transonic shock-free flows continuous in the $(a - M_\infty)$ parameter space can and do exist for our problem. The solid line in Figure 2.6 represents the loci of sonic vortices, i.e., $M_v = 1$, where M_v is the Mach number on the vortex boundary and the circles represent the first occurrence of limit lines.

2.6.1 Evacuated vortex core

The dotted line in Figure 2.6 is an upper boundary on the supersonic flow region. It actually corresponds to a limit in which the pressure in the core of the vortex is reduced to zero. This evacuated core limit is a special case since our problem then reduces to a single parameter family. Recalling the energy equation (2.2), we note the following:

$$\frac{c_v^2}{\gamma-1}+\frac{1}{2}q_v^2=\frac{c_\infty^2}{\gamma-1}+\frac{1}{2}q_\infty^2. \tag{2.37}$$

Since the core is at vacuum, $c_v = 0$, manipulating the above equation gives a relationship for M_∞ in terms of a,

$$M_\infty=\sqrt{\frac{2}{\gamma-1}\frac{a^2}{1-a^2}}. \tag{2.38}$$

In relation to Figure 2.6, this corresponds to the dotted line. We chose a as the parameter for this search and found that limit lines appeared quite early. Actually, the largest value attained before the Jacobian of the transformation changed sign was $a = 0.0015$ corresponding to $M_\infty = 0.00335$. Note that in order to achieve good resolution of the solution near $q = 0$, where we have a Neumann condition, we had to use fine grids. The evacuated vortex results were computed using a resolution of $(N, M + 1) = (2000 \times 128), (4000 \times 128)$ and (4000×256). At this point it was necessary to determine the validity of the two solutions for which there was no change of sign of the Jacobian. Hence, we searched for the minimum of the Jacobian and monitored its value as we increased the resolution. Figure 2.7 clearly shows that as the resolution is increased, the Jacobian tends to zero. This leads us to conclude that there exists no evacuated vortex solution for this problem free of limit lines.

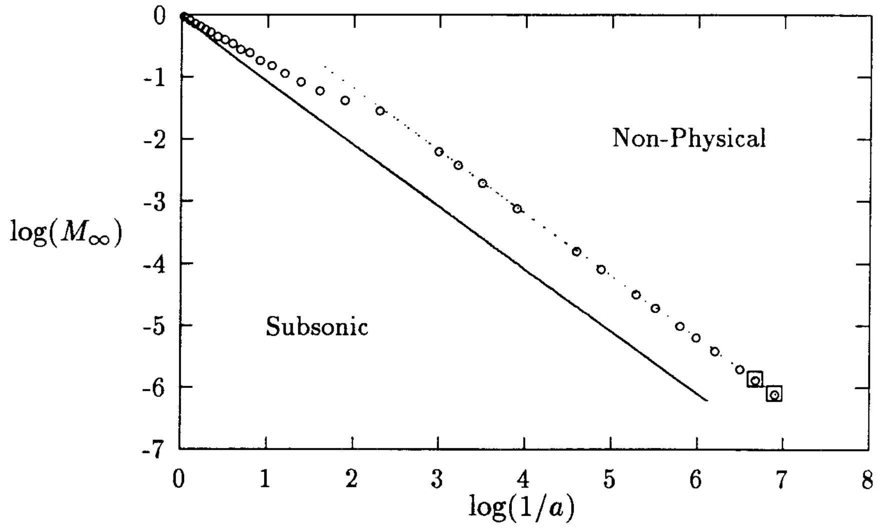


Figure 2.6: The solid line represents the loci of sonic vortices, $M_v = 1.0$. The dotted line indicates the evacuated vortex core limit, $p_v = 0$. The symbol ($\circ$) represents a change of sign of the Jacobian and ($\square$) indicates no change of sign.

Following [18], we find a leading-order approximation to the value of the stream function at the boundary of the evacuated vortex by the method of matched asymptotic expansions (Appendix B). The incompressible solution, [12], is an outer solution which is matched to the solution for an isolated hollow vortex, [26], giving

$$\begin{aligned} \psi_v &= \frac{\Gamma}{4\pi}[F(1;\gamma) - \log a^2)] \\ F(s;\gamma) &= \frac{1}{\gamma-1}\int_0^s \log\xi(1-\xi)^{\frac{2-\gamma}{\gamma-1}}d\xi. \end{aligned} \tag{2.39}$$

M_∞	ψ_v	
	Numerical	Asymptotic
0.225	-5.8946	-5.84959
0.112	-8.6342	-8.62218
0.045	-12.2892	-12.28734
0.022	-15.0600	-15.0599
0.00335	-22.6540	-22.6484

Table 2.2: A comparison of the numerical and asymptotic results for the evacuated vortex.

In Table (2) we compare the asymptotic result for the value of the stream function on the vortex boundary with that from our numerical solution. Even though all evacuated vortex solutions exhibited cuspidal behaviour corresponding to the onset of limit lines, the location of the change of sign of the Jacobian was never on the vortex boundary. Hence, we get good agreement when comparing the result from equation (2.39) with the full numerical solution.

2.6.2 Streamlines and limit lines

While monitoring the Jacobian for a change of sign, we also noted the exact location on the grid for which the limit lines first appear. Figure 2.8 displays a few examples. We have plotted the boundary of the vortex and the position of the first occurrence of a limit line for a given (a, M_∞).

In Figure 2.9, we display the free boundaries for a progression of vortices ranging from incompressible to subsonic to supersonic flow in the range of

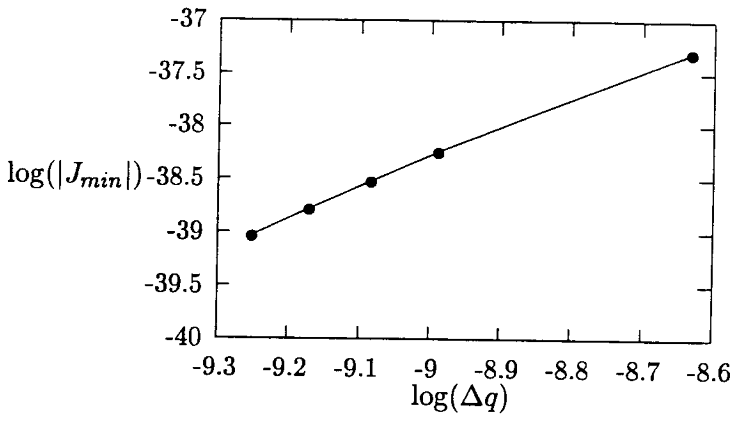


Figure 2.7: The absolute value of the minimum of the Jacobian is plotted versus different mesh sizes. $M = 256$ is kept constant as Δq is decreased.

the (M_∞, a) space where no limit lines were detected. Note that for the free boundaries shown, the shape ratio was held fixed, $a = 0.2$. We see that as M_∞ is increased, the vortices shrink in size and get closer together. Intuitively, one expects compression of the type seen. In addition, there exists some experimental evidence showing a similar effect. Mungal, Hermanson and Dimotakis have produced some unpublished Schlieren data which shows that the large scale structures in a shear layer shrink in size as the flow velocity is increased. For photos and discussion refer to [6]. Note that the present analogy is only qualitative.

In Figure 2.10, we have plotted the streamlines of a transonic flow in the physical plane. The dotted line on the figure denotes the sonic line. Note that the sonic line is closer to the vortex boundary in the x-direction than the y-direction. This phenomenon is explained by the fact that the flow in the x-direction is required to reach a stagnation point.

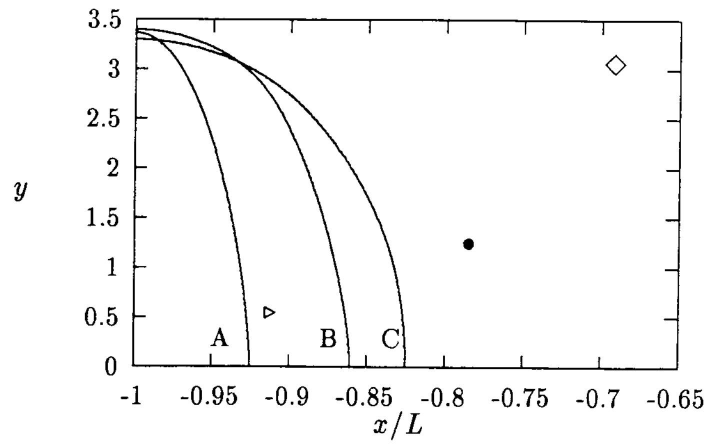


Figure 2.8: Vortex boundaries and the position of the corresponding limit lines. Boundary A and (▷) correspond to $(a, M_\infty) = (0.1, 0.215)$. Boundary B and (•) correspond to $(a, M_\infty) = (0.2, 0.296)$. Boundary C and (⋄) correspond to $(a, M_\infty) = (0.25, 0.343)$.

2.6.3 Vortex geometry

At this point, we return to Figure 2.6 to identify the range of (a, M_∞) for which solutions exist and survey this parameter space for the associated vortex boundary geometries. Figure 2.11 shows the relevant length scales in the physical plane.

Note that b_1 and b_2 are measured from the center of the vortex and they measure the width and height of the vortex boundary respectively. The length λ $(= L/2)$ measures the distance from the center of the vortex to the stagnation point. We found that presenting the aspect ratio of the vortex boundary, b_1/b_2, and the parameter b_1/λ was sufficient to give a complete overall view of all possible geometrical configurations that the vortices attain at different values of (a, M_∞).

Figure 2.12 shows that for a fixed value of a, the vortices shrink and get

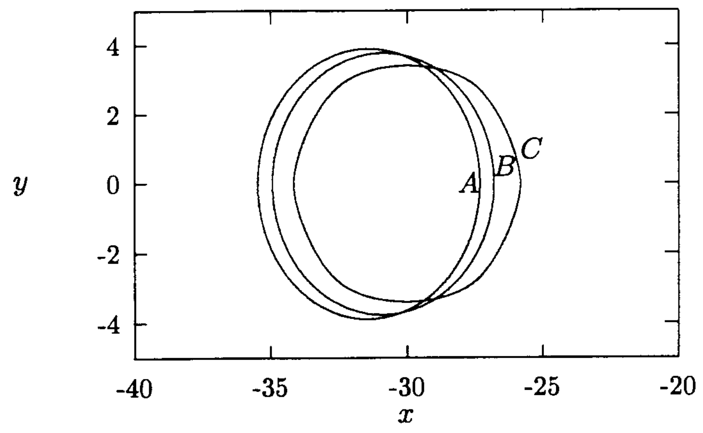


Figure 2.9: Incompressible, sonic and supersonic vortices. $a = 0.2$ is kept fixed and M_∞ is increased. The Mach number on the vortex boundaries is $M_{v_A} = 0.0, M_{v_B} = 1.00$ and $M_{v_C} = 1.933$. The distance x is measured from the stagnation point.

closer together as the Mach number is increased. Note that $b_1/b_2 = 1$ is the limit of circular vortices. Figure 2.13 shows the extent to which the vortex boundary is stretched in the horizontal direction. It is evident that for a fixed a, as the Mach number is increased, the vortex boundary stretches out in the horizontal direction.

When $b_1/\lambda = 0$, we are at the point vortex solution whereas $b_1/\lambda = 1$ is the limit of the shear layer solution. Note that in both Figure 2.12 and Figure 2.13, the solutions were terminated at the first occurrence of a limit line.

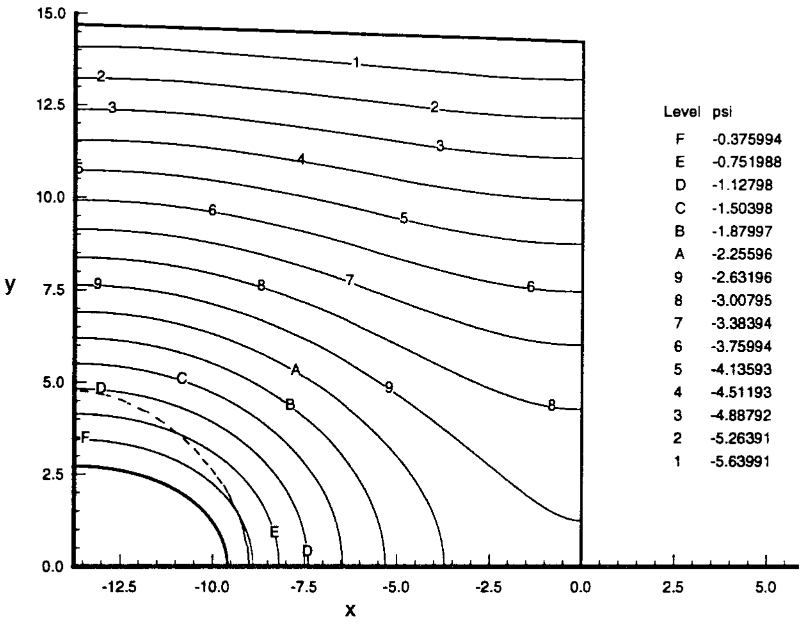


Figure 2.10: Streamlines for a transonic shock-free flow. $a = 0.4$ and M_∞=0.48. Here, $M_v = 1.4$, where M_v is the Mach number on the vortex boundary indicating supersonic flow. The dotted line indicates the sonic line.

.

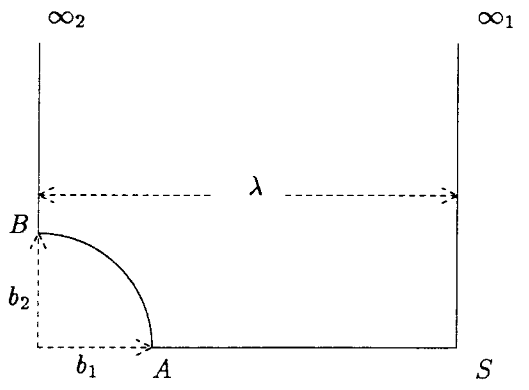


Figure 2.11: A description of the relevant length scales of the problem.

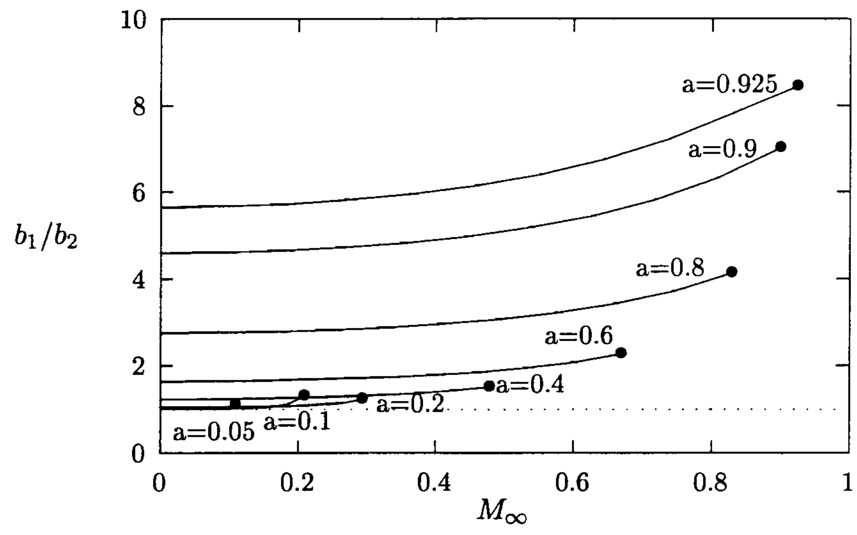


Figure 2.12: Aspect ratio of the vortex boundary for all possible solutions. The dotted line represents the limit of circular vortices. (●) indicates the occurrence of a limit line.

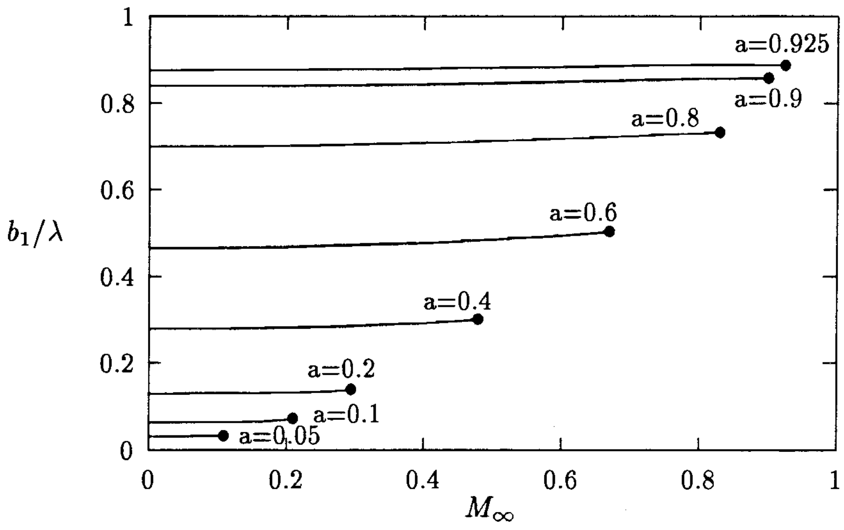


Figure 2.13: Stretching of the vortex boundary in the x-direction. (●) indicates the occurrence of a limit line.

Chapter 3

Compressible Stuart-type vortices

For two dimensional incompressible flow, it can be shown that the vorticity equation is satisfied by taking $\omega = F(\psi)$. J.T. Stuart [25], while investigating mixing layers of the form $u = \tanh y$, realized that by taking

$$\omega = B\exp(-2\psi/A),$$

the governing equation for incompressible flows with this specific form of vorticity distribution leads to an example of the well-known Liouville equation [3]

$$\nabla^2\psi = B\exp(-2\psi/A), \tag{3.1}$$

where A and B are constants. Taking $B = 2\pi u_\infty/\lambda$ and $A = u_\infty\lambda/2\pi$, an exact solution of equation (3.1) can be written as

$$\psi(x, y) = \frac{u_\infty\lambda}{2\pi} \log\left[\kappa \cosh\frac{2\pi y}{\lambda} + \sqrt{\kappa^2 - 1}\cos\frac{2\pi x}{\lambda}\right]. \qquad (3.2)$$

This solution describes an infinite array of vortices of circulation $u_\infty\lambda/2\pi$, lying on the x-axis, separated a fixed distance λ. The constant κ is a parameter with the range $1 \leq \kappa \leq \infty$ that measures the compactness of the vorticity distribution. The limit $\kappa \rightarrow 1$ gives a parallel shear flow. The limit $\kappa \rightarrow \infty$ describes the flow of an infinite array of point vortices in an unbounded fluid. At this stage we choose to rescale the parameter κ as suggested by [15],

$$\epsilon = \sqrt{\kappa^2 - 1}/\kappa \qquad (3.3)$$

such that $0 \leq \epsilon \leq 1$, and hence the vorticity distribution can be written as

$$\omega = \frac{2\pi u_\infty}{\lambda}(1 - \epsilon^2)\exp\left[-\frac{4\pi\psi}{u_\infty\lambda}\right]. \qquad (3.4)$$

The Stuart vortex solutions then take the form,

$$\psi(x, y) = \frac{u_\infty\lambda}{2\pi} \log\left[\cosh\frac{2\pi y}{\lambda} + \epsilon\cos\frac{2\pi x}{\lambda}\right]. \qquad (3.5)$$

The significance of the above transformation is to introduce the parameter ϵ in the expression for the vorticity so as to facilitate the extension of the Stuart vortex solutions to compressible flow. This will become more evident as we derive the governing equations. We shall denote flows with the vorticity distribution given by (3.4) as Stuart-type vortices. Before we proceed to find Stuart-type vortices in a channel, we need to discuss the asymptotics of this very special solution of the inviscid Euler equations. We shall denote the stream function at infinity as ψ_∞, and by taking the limit $y \to \infty$ in equation (3.5), we can identify the nature of the singularity of the stream function.

$$\psi_\infty \sim \frac{u_\infty \lambda}{2\pi} [y + \log \frac{1}{2} + O(2\,\epsilon\, \cos x\, \exp(-y))]. \tag{3.6}$$

Note that the transformation suggested by equation (3.3) has yet another useful feature in that it fixes the constant at infinity given in equation (3.6) as log(1/2). Had we employed κ as a parameter, the correction to the singularity at infinity for the stream function would have been a function of κ as is shown below:

$$\psi_\infty \sim \frac{u_\infty \lambda}{2\pi} [y + \log \frac{\kappa}{2} + O(\sqrt{\kappa^2 - 1}\, \cos x\, \exp(-y))].$$

3.1 Generalized Stuart-vortex solutions

We can write the Stuart vortex solution in nondimensional form as

$$\psi(x, y) = \log [\cosh y + \epsilon \cos x]$$

and calculate its circulation, Γ, in the domain depicted in Figure 3.2 with the exception that the upper boundary is at $y \to \infty$. We find that

$$\begin{aligned} \Gamma &= \oint \vec{u} \cdot \vec{dl} \\ &= \int_0^\pi u_\infty dx = \pi. \end{aligned} \tag{3.7}$$

We now ask whether this solution is unique. In other words, we will try to construct solutions to

$$\nabla^2\psi = (1-\epsilon^2)\exp(-2\psi), \tag{3.8}$$

with the same distribution of vorticity as that of the Stuart vortex solution that have different circulations. We shall assume

$$\psi(x,y) = \log\left[f(y) + \epsilon g(x)\right] \tag{3.9}$$

and substitute this into the left-hand side of equation (3.8). After some algebra, we have

$$\nabla^2\psi = [(f''f - f'^2) + (g''f + gf'')\epsilon + (g''g - g'^2)\epsilon^2]/(f+\epsilon g)^2, \tag{3.10}$$

and since we require the same vorticity distribution as the Stuart solution, we match the right-hand side of equation (3.8) with equation (3.10) to get the conditions on our unknown functions $f(y)$ and $g(x)$;

$$f''f - f'^2 = 1, \tag{3.11}$$

$$g''f + gf'' = 0, \tag{3.12}$$

$$g''g - g'^2 = -1. \tag{3.13}$$

Solving (3.12) we find that

$$f(y) = C\exp(\alpha y) + D\exp(-\alpha y), \tag{3.14}$$

$$g(x) = A\cos(\alpha x) + B\sin(\alpha x), \tag{3.15}$$

where A, B, C, D and α are constants of integration determined by satisfying equations (3.11) and (3.12):

$$\begin{aligned} 4C\alpha^2 D &= 1, & (3.16)\\ \alpha^2(A^2+B^2) &= 1. & (3.17)\end{aligned}$$

Note that α is assumed real. Using the above conditions we can write a solution to equation (3.8) in the following form:

$$\psi(x,y) = \log\left[C\exp(\alpha y) + \frac{1}{4\alpha^2 C}\exp(-\alpha y) + \epsilon\left[A\cos(\alpha x) \pm \frac{\sqrt{1-\alpha^2A^2}}{\alpha}\sin(\alpha x)\right]\right]. \quad (3.18)$$

The circulation of this solution is given by

$$\begin{aligned}\Gamma &= \oint \vec{u}\cdot\vec{dl},\\ &= \int_0^{\frac{\pi}{\alpha}} u_\infty dx = \pi. \quad (3.19)\end{aligned}$$

Yet, equation (3.18) can always be mapped back onto the Stuart vortex solution through the appropriate stretching and translation of the coordinate axes. The mapping from the Stuart vortex to the solution given by equation (3.18) is given by the following:

$$y = \alpha\hat{y} + \log(2\alpha C),$$

$$x = \alpha\hat{x} + \arccos(\alpha A).$$

The derivation of the Stuart vortex solution given by equation (3.18) shows that as $y \to \infty$, then

$$\psi_\infty \sim \alpha y + \log(C/2).$$

The usefulness of the above result will be discussed further in chapter 4. We now proceed to derive the equations governing the compressible Stuart-type vortices.

3.2 Governing equations

The inviscid flow of a compressible fluid is described by the Euler equations:

$$\frac{\partial \rho}{\partial t} + (\mathbf{u} \cdot \nabla)\rho + \rho \nabla \cdot \mathbf{u} = 0, \tag{3.20}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p. \tag{3.21}$$

We shall be considering two dimensional, steady flow, and hence we can define a stream function by the relations

$$\begin{aligned} \rho u &= \rho_{ref}\frac{\partial \psi}{\partial y}, \\ \rho v &= -\rho_{ref}\frac{\partial \psi}{\partial x}, \end{aligned} \tag{3.22}$$

where ρ_{ref} is some reference density, such that the continuity equation (3.20) is automatically satisfied. Using the definition of vorticity ($\omega = \nabla \times \mathbf{u}$) along with equation (3.22), we can relate the stream function of a compressible flow to the vorticity distribution by the elliptic equation

$$(\frac{\rho_{ref}}{\rho}\psi_x)_x + (\frac{\rho_{ref}}{\rho}\psi_y)_y = -\omega. \tag{3.23}$$

In order to have a closed set of equations, we need to specify the thermodynamics of the flow. We use the result from the first and second law of thermodynamics:

$$Tds = dH - dp/\rho, \tag{3.24}$$

where T is the temperature, s, the entropy, H, the enthalpy, and p, the pressure of the gas, in conjunction with the momentum equation (3.21) to arrive at a useful relationship known as Crocco's Theorem [5]. In its general form, Crocco's Theorem can be written as

$$T\nabla s = \nabla H_0 - \mathbf{u} \times (\nabla \times \mathbf{u}) + \frac{\partial \mathbf{u}}{\partial t}. \tag{3.25}$$

This gives a relationship between the kinematics of the flow and the thermodynamics of the fluid. Note that for steady flow, the above result relates the vorticity in the flow to the thermodynamics of the fluid motion:

$$\mathbf{u} \times \omega = \nabla H_0 - T\nabla s. \tag{3.26}$$

Here $\omega = \nabla \times \mathbf{u}$ is the vorticity and $H_0 = H + u^2/2$ is the total enthalpy of the flow. There are two limits to equation (3.26) that are relevant to our problem. The constraint $\nabla s = 0$ corresponds to homentropic flow. The constraint $\nabla H_0 = 0$ corresponds to homenthalpic flow. The latter constraint is of interest because it will allow for a change in the entropy distribution throughout the flow and, hence, might lead to the formation of shock waves.

Before proceeding with our problem, we recall the vorticity equation for compressible flow in two dimensions:

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla)\, \omega + \omega\, (\nabla \cdot \mathbf{u}) = \frac{1}{\rho^2} \nabla \rho \times \nabla p. \tag{3.27}$$

We shall need to make use of the definition of the total derivative, $D/Dt = \partial/\partial t + \mathbf{u} \cdot \nabla$. Using the chain rule for differentiation in conjunction with the vorticity and continuity equations, we arrive at the following compact form for the vorticity:

$$\frac{D}{Dt}(\omega/\rho) = \frac{1}{\rho^3} \nabla \rho \times \nabla p. \tag{3.28}$$

The importance of this form of the vorticity equation shall be more evident as we further discuss the type of solutions we wish to capture.

3.2.1 Homentropic flow

A flow in which the entropy is constant *everywhere* in the flow is said to be a homentropic flow. It can be shown that the equation of state for such a flow reduces to the following:

$$\frac{p}{\rho^\gamma} = \mathit{constant}, \tag{3.29}$$

where $\gamma = c_p/c_v$ is the ratio of specific heats. This implies that the gradient of density and pressure are aligned and hence the right-hand side of equation (3.28) is identically zero. Hence, for two dimensional, steady, homentropic flow of a compressible fluid, we can state that the ratio of the vorticity to the local density is constant along a particle path,

$$\frac{D}{Dt}(\omega/\rho) = 0. \tag{3.30}$$

Since particle paths are streamlines for steady flow, solutions of (3.30) can be written in the following form:

$$\frac{\omega}{\rho} = -\frac{1}{\rho_{ref}} F(\psi). \tag{3.31}$$

The left-hand side of equation (3.26) can be re-written as

$$\begin{aligned} \mathbf{u} \times \omega &= \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ (\rho_{ref}/\rho)\,\psi_y & -(\rho_{ref}/\rho)\,\psi_x & 0 \\ 0 & 0 & -(\rho/\rho_{ref})\,F(\psi) \end{vmatrix} \\ &= \psi_x F(\psi)\,\hat{i} + \psi_y F(\psi)\,\hat{j} \\ &= \nabla G, \end{aligned} \tag{3.32}$$

where

$$G = \int_{\psi_{ref}}^{\psi} F(\eta) d\eta.$$

The assumption of homentropic flow in conjunction with the definition of total enthalpy and the result above leads to

$$\nabla \left[H + \frac{1}{2}(u^2 + v^2) - G \right] = 0,$$

or

$$H + \frac{1}{2}(u^2 + v^2) - G = constant. \tag{3.33}$$

We shall be solving for the flow of an ideal gas; hence,

$$H = c_p T = \gamma RT/(\gamma - 1) = \frac{a^2}{\gamma - 1}. \tag{3.34}$$

Furthermore, the assumption of homentropic flow then gives a relationship between the speed of sound of the fluid and the density distribution so that we can write

$$a^2 = \gamma RT = \gamma RT_{ref}\,(\frac{T}{T_{ref}}) = a^2_{ref}\,(\frac{\rho}{\rho_{ref}})^{\gamma-1}. \qquad (3.35)$$

Combining equations (3.33), (3.34) and (3.35) allows us to rewrite the energy equation in the following form:

$$\frac{a^2_{ref}}{\gamma-1}(\frac{\rho}{\rho_{ref}})^{\gamma-1} + \frac{1}{2}(\frac{\rho_{ref}}{\rho})^2\,(\psi_x^2+\psi_y^2) + \int_{\psi}^{\psi_{ref}} F(\zeta)\,d\zeta = constant. \quad (3.36)$$

Equations (3.23), (3.31) and (3.36) describe the steady flow of a homentropic, compressible flow in two dimensions. At this stage, we can specify a desired vorticity distribution and hence reduce our system to two equations for two unknowns, namely the density and the stream function. The choice of vorticity distribution is arbitrary, but we choose to investigate those types of flow that have the same distribution of vorticity as that suggested by [25].

3.3 Stuart-type vortices with walls

Equation (3.6) shows that the stream function of the incompressible Stuart vortex in an unbounded domain ($y \to \infty$) is singular. We choose to investigate Stuart-type vortices in a channel of finite height as depicted in Figure 3.1 so as to understand the nature of this singularity for compressible flow thru the limit $h \to \infty$.

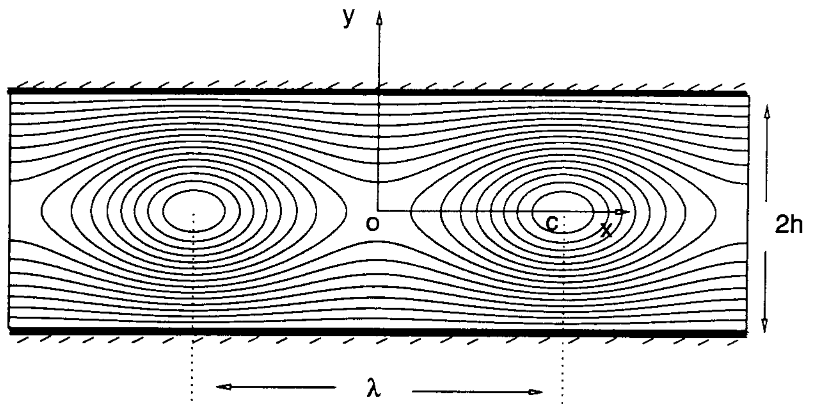
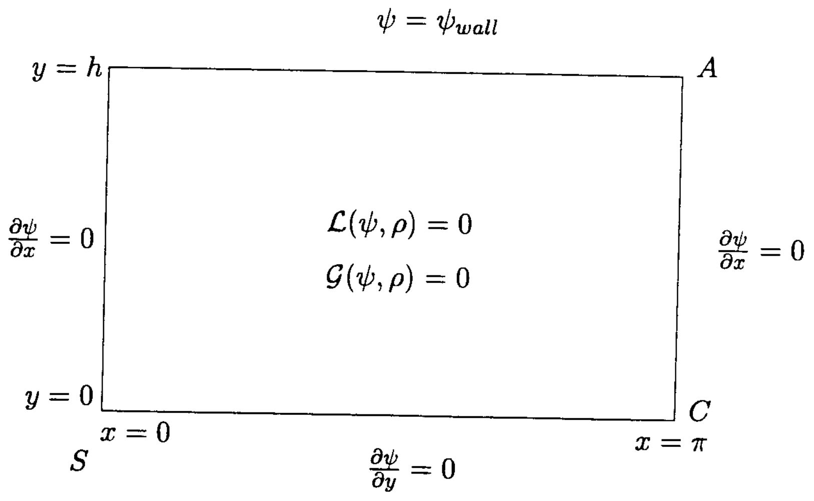


Figure 3.1: A sketch of the streamlines for Stuart-type vortices with walls. The origin of the coordinate system is positioned at the stagnation point of the flow. The channel height is $2h$. Point C denotes the vortex core.

The vortices are of Stuart-type because we require the vorticity distribution throughout the flow to be of the form given in equation (3.4). We assume that the flow is periodic in the x-direction with period λ and hence construct the solutions to our problem in a quarter of the geometry depicted in Figure 3.1. The right- hand side boundary of the computational domain is constructed by drawing a line through the vortex core (point C in Figure 3.2) parallel to the y-axis and denoting the intersection of this line with the wall as point A. The computational domain along with the relevant boundary conditions is depicted in Figure 3.2.

We shall choose the conditions at point A, depicted in Figure 3.2, as the reference velocity and density. Since the flow is inviscid, the wall is a stream function of the flow and hence its value, ψ_{wall}, is constant along the



$\psi = \psi_{wall}$
$y = h$
A
$\frac{\partial \psi}{\partial x} = 0$
$\mathcal{L}(\psi, \rho) = 0$
$\mathcal{G}(\psi, \rho) = 0$
$\frac{\partial \psi}{\partial x} = 0$
$y = 0$
C
$x = 0$
S
$\frac{\partial \psi}{\partial y} = 0$
$x = \pi$

Figure 3.2: Computational domain and boundary conditions of the problem. The value of the streamfunction on the wall is a constant, ψ_{wall}. Point S is the position of the stagnation point. Point C locates the vortex core.

line $y = h$. With point A as our choice of reference state, we can rewrite equations (3.23), (3.31) and (3.36) as follows:

$$(\frac{\rho_A}{\rho}\psi_x)_x + (\frac{\rho_A}{\rho}\psi_y)_y = \frac{\rho}{\rho_A}\mathcal{F}(\psi), \tag{3.37}$$

$$\frac{a_A^2}{\gamma - 1}(\frac{\rho}{\rho_A})^{\gamma-1} + \frac{1}{2}(\frac{\rho_A}{\rho})^2(\psi_x^2 + \psi_y^2) + \int_{\psi}^{\psi_{wall}} F(\zeta)\, d\zeta = constant, \tag{3.38}$$

$$\mathcal{F}(\psi) = \frac{2\pi u_A}{\lambda}(1 - \epsilon^2)\exp(-\frac{4\pi\psi}{u_A\lambda}). \tag{3.39}$$

The value of the constant in equation (3.38) is determined by evaluating the equation at the reference state A. We note that the vorticity term in the energy equation (3.38) evaluated at the wall is identically zero, and hence the value of the constant can be related to the conditions at point A.

$$constant = \frac{a_A^2}{\gamma - 1} + \frac{u_A^2}{2} \equiv C_A. \tag{3.40}$$

Before proceeding with the solution, we need to find the appropriate scalings for the problem and determine the number of relevant physical parameters.

3.3.1 Scaling of variables

We shall define the Mach number at the reference point A as $M_A = u_A/a_A$ and evaluate the integral term in equation (3.38) analytically. Furthermore, we define the following non-dimensional quantities;

$$\begin{aligned} \hat{x} = \frac{2\pi x}{\lambda}, &\quad \hat{y} = \frac{2\pi y}{\lambda}, \\ \hat{\rho} = \frac{\rho}{\rho_A}, &\quad \hat{\psi} = \frac{2\pi\psi}{u_A\lambda}, \end{aligned} \tag{3.41}$$

and proceed to solve the following equations on the computational grid depicted in Figure 3.2:

$$(\frac{1}{\hat{\rho}}\hat{\psi}_{\hat{x}})_{\hat{x}} + (\frac{1}{\hat{\rho}}\hat{\psi}_{\hat{y}})_{\hat{y}} = \hat{\rho}\,(1 - \epsilon^2)\,\exp(-2\hat{\psi}), \tag{3.42}$$

$$\begin{aligned} \hat{\rho}^{\gamma-1} &- (1 + \frac{\gamma-1}{2}M_A^2) + \frac{\gamma-1}{2}M_A^2\frac{1}{\hat{\rho}^2}\,(\hat{\psi}_{\hat{x}}^2 + \hat{\psi}_{\hat{y}}^2) \\ &= -\frac{\gamma-1}{2}M_A^2\,(1-\epsilon^2)\,[\exp(-2\hat{\psi}) - \exp(-2\hat{\psi}_{wall})]. \end{aligned} \tag{3.43}$$

We choose to omit the use of the hat notation in what follows below. It should be noted that all variables used will be non-dimensional unless otherwise stated.

In the limit of infinite sound speed, the Mach number tends to zero and the energy equation (3.43) gives $\rho = 1$ everywhere in the flow. Substituting this result into equation (3.42) then gives us Louiville's equation for which we know Stuart vortices are an exact solution. It is possible to recover the Stuart vortex solution by an appropriate choice of boundary conditions on the wall. The asymptotics of the Stuart vortex solution as given by equation (3.6) suggests that if $\psi_{wall} = \log[\cosh(h)]$ and $M_A = 0$, then in the limit as $h \to \infty$, the solutions to equations (3.42) and (3.43) shall approach the Stuart vortex solution.

3.4 Numerical Method

We choose to stretch the vertical axis, $y' = y/h$, so that the vertical axis of the domain depicted in Figure 3.2 transforms to $0 \le y' \le 1$. This stretching is also applied to Equations (3.42) and (3.43) and, hence, the wall height, h, now appears as a parameter in our system of equations.

$$(\frac{1}{\rho}\psi_x)_x + \frac{1}{h^2}(\frac{1}{\rho}\psi_{y'})_{y'} = \rho\,(1-\epsilon^2)\,\exp(-2\psi), \tag{3.44}$$

$$\begin{aligned}\rho^{\gamma-1} &- (1+\frac{\gamma-1}{2}M_A^2) + \frac{\gamma-1}{2}M_A^2\frac{1}{\rho^2}\,(\psi_x^2 + \frac{1}{h^2}\psi_{y'}^2) \\ &= -\frac{\gamma-1}{2}M_A^2\,(1-\epsilon^2)\,[\exp(-2\psi) - \exp(-2\psi_{wall})].\end{aligned} \tag{3.45}$$

The nonlinear nature of the above equations requires an iterative numerical procedure. As in Newton's method, we write the stream function and the density as an initial guess plus some correction

$$\psi = \psi_0 + \delta\psi,$$

$$\rho = \rho_0 + \delta\rho,$$

and substitute the above expressions into equations (3.44) and (3.45). Linearizing the resulting system in $\delta\psi$ and $\delta\rho$ leads to a linear system given in Appendix C. ψ_0 and ρ_0 are the initial guesses, and so we attempt to approach the correct solution by solving for $\delta\psi$, $\delta\rho$ and then updating ψ and ρ. It is important to note here that equation (3.45) is really a scalar equation in the variable ρ, and so it is possible to analytically solve for $\delta\rho$. This reduces the size of the Jacobian by half since the matrix that needs to be inverted at each iteration step now only corresponds to the unknowns $\delta\psi$.

Second order finite difference formulas are used to evaluate the coefficients of the Jacobian, and hence the discretized system of equations

$$\mathbf{A}_{i,j} \cdot \delta\psi_{i,j} = \mathbf{B}_{i,j} \tag{3.46}$$

is solved on the grid defined by

$$\begin{aligned} x_i &= (i-1)\,dx, \qquad & i &= 1, N \\ y_j' &= (j-1)\,dy, \qquad & j &= 1, M \end{aligned} \tag{3.47}$$

where $dx = \pi/(N-1)$ and $dy' = 1/(M-1)$, subject to the modified boundary conditions

$$\begin{aligned}
\frac{\partial(\delta\psi)}{\partial y'} &= -\frac{\partial\psi_0}{\partial y'} \qquad & y' = 0, \\
\frac{\partial(\delta\psi)}{\partial x} &= -\frac{\partial\psi_0}{\partial x} \qquad & x = 0,\ \pi \\
\delta\psi &= \psi_{wall} - \psi_0 \qquad & y' = 1.
\end{aligned} \tag{3.48}$$

The entries of the matrices $\mathbf{A}_{i,j}$ and $\mathbf{B}_{i,j}$ are given in Appendix C. For a given Mach number M_A, the parameters that need to be fixed are h, ψ_{wall}, and ϵ. If we set $\psi_{wall} = \log[\cosh(h)]$, $M_A = 0$ and $\epsilon = 0.5$ and take the limit of equations (3.44) and (3.45) as $h \to \infty$ numerically, we must recover the incompressible Stuart vortex solution given by equation (3.5) for $\epsilon = 0.5$. This is the main reason for the transformation from κ to ϵ as suggested in equation (3.3). If we had used κ as a parameter, there would be no clear limiting procedure to recover the unbounded, incompressible Stuart vortex solutions from the compressible Stuart-type vortices with walls. Figure (3.3) shows that the result from the calculations of $h = 6$ is in good agreement with the incompressible, unbounded Stuart vortex solution given by equation (3.5). This suggests that $h = 6$ is a good approximation for $h \to \infty$.

3.5 Perturbation solution

We construct an approximate series solution to equations (3.44) and (3.45) valid when $M_A \ll 1$. The solution we seek has the Rayleigh-Janzen form,

$$\psi(x,y) \;=\; \psi_0(x,y) + M_A^2\, \psi_1(x,y) + O(M_A^4),$$

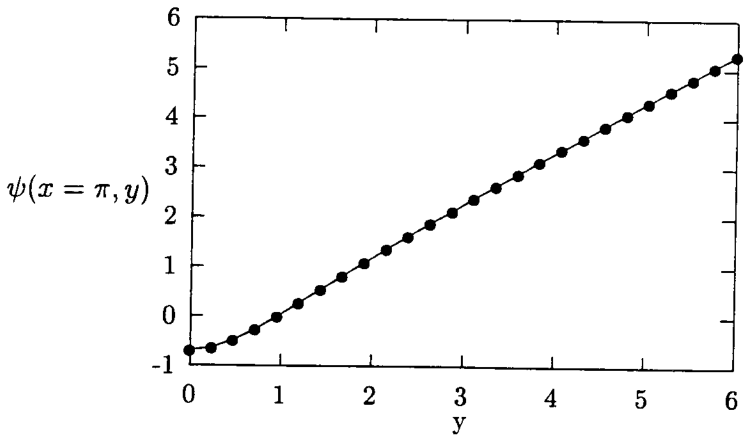


Figure 3.3: Comparison of Stuart Vortex solution as given by equation 3.5 and numerical result for $M_A = 0$, $\epsilon = 0.5$, $\psi_{wall} = \log[\cosh(h)]$ and $h = 6$. The circles represent the numerical solution and the solid line depicts the analytical result.

$$\rho(x,y) \quad = \quad 1 + M_A^2\, \rho_1(x,y) + O(M_A^4). \tag{3.49}$$

Substituting the above series form into equations (3.44) and (3.45) gives an equation for the zeroth order incompressible solution

$$\psi_{0_{xx}} + \frac{1}{h^2}\psi_{0_{yy}} = (1-\epsilon^2)\exp(-2\psi_0), \tag{3.50}$$

along with the following boundary conditions:

$$\begin{aligned} \psi_{0_x} &= 0 \qquad & x &= 0, \pi \\ \psi_{0_{y'}} &= 0 \qquad & y' &= 0, \\ \psi_0 &= \psi_{wall} \qquad & y' &= 1. \end{aligned} \tag{3.51}$$

The zeroth order solution must be obtained numerically since we do not have an analytical form for the Stuart vortex flow with walls. Note that this analysis shows that $\rho_0(x,y) = 1$ is the correct solution to this order.

Since equation (3.45) is algebraic in $\rho(x,y)$, it is possible to obtain an analytic expression for the density distribution from the second order problem,

$$\rho_1(x,y) = \frac{1}{2}\,(1 - (\frac{\partial\psi_0}{\partial x})^2 - \frac{1}{h^2}(\frac{\partial\psi_0}{\partial y'})^2 - (1-\epsilon^2)[\exp(-2\psi_0) - \exp(-2\psi_{wall})]\,, \tag{3.52}$$

in terms of the known incompressible solution. This is not the case for the stream function which requires numerical solution of the equation

$$\begin{aligned} \psi_{1_{xx}} + \frac{1}{h^2}\psi_{1_{y'y'}} + 2(1-\epsilon^2)\exp(-2\psi_0)\,\psi_1 &= \rho_1(1-\epsilon^2)\exp(-2\psi_0) \\ &+ (\rho_1\psi_{0_x})_x \\ &+ \frac{1}{h^2}(\rho_1\psi_{0_{y'}})_{y'}. \end{aligned} \tag{3.53}$$

For this analysis, we assume that the value of the stream function on the wall, ψ_{wall}, has no Mach number dependence. This does not have to be the case. Since we are constructing the perturbation solutions to serve as a check for the full numerical algorithm, it is only necessary to be consistent when it comes to the definition of ψ_{wall}. Hence, the boundary conditions consistent with this assumption for the second order problem are as follows:

$$\begin{aligned} \psi_{1_x} &= 0 \qquad x = 0, \pi \\ \psi_{1_{y'}} &= 0 \qquad y' = 0, \\ \psi_1 &= 0 \qquad y' = 1. \end{aligned} \tag{3.54}$$

y	$\psi(x=\pi, y')$	
	Numerical	Perturbation
0.00	-0.634237	-0.634791
0.20	-0.487040	-0.487600
0.40	-0.121898	-0.122383
0.60	0.338631	0.338302
0.80	0.827671	0.827511
1.00	1.325003	1.325003

Table 3.1: A comparison of the numerical and perturbation results at $M_A = 0.1$, $h = 2$, $\psi_{wall} = \log\cosh h$, and $\epsilon = 0.5$.

The compressibility correction to the zeroth order incompressible problem of Stuart-type vortices with walls is found numerically. Combining the incompressible solution with the compressible corrections as suggested by equation (3.49) then provides an approximate solution that can be compared to the full numerical solution discussed in the previous section. We test the numerical algorithm by performing both the full numerical simulation and calculating the perturbation solution for $M_A = 0.1$, $h = 2$, and $\psi_{wall} = \log[\cosh h)]$. The results are shown in tables (3.1) and (3.2) for both the density distribution and the stream function. Since the perturbation solution is accurate to $O(M_A^4)$ and since $M_A = 0.1$, we would expect any difference between the full numerical solution and the perturbation result to be in the fourth significant digit which is indicated by Tables (3.1) and (3.2).

x	$\rho(x, y^{'} = 0)$	
	Numerical	Perturbation
0.000000	1.002749	1.002880
0.628319	1.002323	1.002458
1.256637	1.000894	1.001031
1.884956	0.998105	0.998187
2.513274	0.994193	0.994065
3.141593	0.991952	0.991629

Table 3.2: A comparison of the numerical and perturbation results at $M_A = 0.1$, $h = 2$, $\psi_{wall} = \log\cosh h$, and $\epsilon = 0.5$.

3.6 Results

We begin by exploring the Mach number dependence of the circulation of the flow as $h \to \infty$. We set $\psi_{wall} = \log[\cosh(h)]$ and $M_A = 0$ and compute the solutions to equations (3.44) and (3.45) for $\epsilon = 0.5$ and increasing h. The computations are done on an $(N, M) = (101, 101)$ grid. The starting value of the wall height is $h = 2$, and the computations are continued until a wall height of $h = 12$. This procedure is then repeated for $M_A = 0.05$ and $M_A = 0.1$. The circulation in the computational domain is shown plotted versus the wall height in Figure 3.4. It is evident that for $M_A = 0$, the circulation is asymptotically approaching $\Gamma = \pi$, the value expected from the Stuart vortex solution. Also, as the Mach number is increased, the circulation in the domain asymptotes to a constant other than $\Gamma = \pi$, suggesting that the circulation for the compressible Stuart-type vortices in

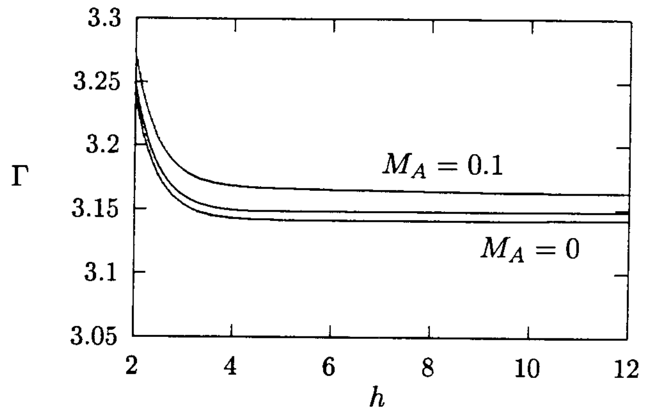


Figure 3.4: The circulation in the domain, Γ, is measured as a function of the wall height, h, for various values of the Mach number, $M_A = (0, 0.05, 0.1)$.

an unbounded domain is a definite function of the Mach number. This was alluded to earlier in section 3.3.1. In the limit as $h \to \infty$, we would expect that the velocity distribution tends to a constant value along the line $y = h$ since the flow at infinity should resemble that of a vortex sheet. Figure 3.5 shows the velocity of the wall reaching a constant in the limit of increasing wall height for $M_A = 0.1$.

3.6.1 Perturbation expansion for $h \to \infty$

Before proceeding any further, we wish to construct yet another check of the numerical results. The asymptotic behaviour of the Stuart vortex solution is given by equation (3.6). It is clear that the value of the stream function on the wall is infinite as $h \to \infty$, hence we can re-write equations (3.41) and (3.42) in the limit as $\psi_{wall} \to \infty$,

$$\left[\frac{1}{\rho}\psi_x\right]_x + \left[\frac{1}{\rho}\psi_y\right]_y = \rho\,(1 - \epsilon^2)\,\exp(-2\psi), \qquad (3.55)$$

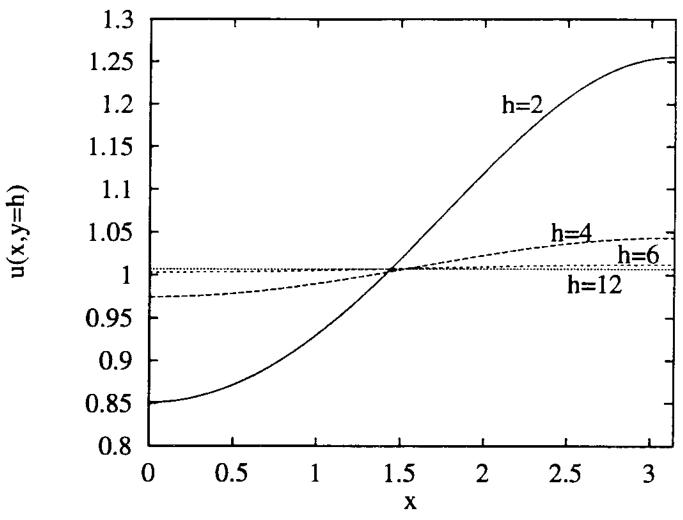


Figure 3.5: Velocity distribution on the wall plotted at different values of the wall height, h. $M_A = 0.1$, $\epsilon = 0.5$ and $\psi_{wall} = \log(\cosh(h))$.

$$\rho^{\gamma-1} + \frac{\gamma-1}{2} M_\infty^2 \frac{1}{\rho^2}\left[\psi_x^2 + \psi_y^2\right] + \frac{\gamma-1}{2} M_\infty^2 (1-\epsilon^2)\exp(-2\psi) = 1 + \frac{\gamma-1}{2} M_\infty^2 \tag{3.56}$$

where

$$\lim_{h\to\infty} M_A^2 = M_\infty^2$$

since the flow at infinity is that which would be produced by a vortex sheet of finite strength.

Now we attempt to construct a perturbation solution for the compressible Stuart-type vortices in an unbounded domain by assuming an expansion of the form

$$\begin{aligned} \psi(x,y) &= \psi_0(x,y) + M_\infty^2\,\psi_1(x,y) + M_\infty^4\,\psi_2(x,y) + \ldots \\ \rho(x,y) &= 1 + M_\infty^2\,\rho_1(x,y) + M_\infty^4\,\rho_2(x,y) + \ldots \end{aligned} \tag{3.57}$$

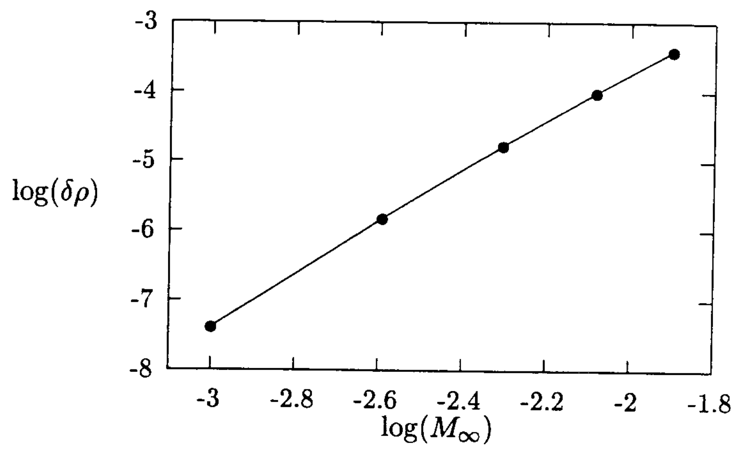


Figure 3.6: The difference between the full numerical solution and the perturbation solution is denoted by $\delta\rho$. The slope of the solid line indicates that the difference between the full and approximate solution is $O(M_\infty^4)$. In this calculation, $\epsilon = 0.8$, $h = 6$

where ψ_0 is that given by equation (3.5). Substituting the above expansions into equations (3.55) and (3.56), we find that since the energy equation is algebraic in ρ, it is easy to solve for the first correction of the density, ρ_1, analytically. Therefore, we can write the density distribution in the flow as

$$\rho(x,y) = 1 + M_\infty^2 \frac{\epsilon \cos x}{\cosh y + \epsilon \cos x}\,. \tag{3.58}$$

We will compare the above result with the numerical calculations for the case of $h = 6$, $\epsilon = 0.8$, $\psi_{wall} = \log[\cosh(h)]$ and $M_A = 0.10$. In Figure (3.6), $\delta\rho$ is defined as the difference between the full numerical solution and the perturbation result. As expected, this difference is $O(M_\infty^4)$. It must be noted here that since obtaining ρ_1 for the compressible Stuart vortex in an unbounded domain is easy (as shown above), it was attempted so as to have yet another analytic result by which to validate the full numerical solution of equations (3.44) and (3.45). Obtaining ψ_1 must be done numerically and it is not a trivial matter. A discussion of the appropriate solution procedure for obtaining compressible Stuart vortex solutions in an unbounded domain is presented in chapter 4.

3.6.2 Parallel shear flow

The parallel shear flow is an interesting limit of the Stuart vortex solution that corresponds to $\epsilon = 0$. In this limit, there is no x-dependence in the flow and hence the governing equations can be simplified and re-written as,

$$\frac{1}{h^2}\left[\frac{1}{\rho}\psi_{y'}\right]_{y'} = \rho(1-\epsilon^2)\exp(-2\psi), \qquad (3.59)$$

$$\begin{aligned}\rho^{\gamma-1} &- \left[1+\frac{\gamma-1}{2}M_A^2\right] + \frac{\gamma-1}{2}M_A^2\frac{1}{\rho^2}\psi_{y'}^2 \\ &= -\frac{\gamma-1}{2}M_A^2(1-\epsilon^2)\left[\exp(-2\psi)-\exp(-2\psi_{wall})\right].\end{aligned} \qquad (3.60)$$

By analogy with one dimensional homentropic flow, if we choose to search for solutions in which the density is a function of the Mach number, M_A, only, we find an exact solution to the above system of equations given below.

$$\begin{aligned}\psi(y') &= \log\left[\cosh\rho\, h\, y'\right], \\ \rho &= (1+\frac{\gamma-1}{2}M_A^2\frac{1}{\cosh^2\rho\, h})^{1/(\gamma-1)}\end{aligned} \qquad (3.61)$$

It is now possible to use the above solution as yet another check on the numerical algorithm. We set $\psi_{wall} = \log(\cosh\rho\, h)$, $\epsilon = 0$ and $h = 2$ and continue the solutions from $M_A = 0$ to $M_A = 0.5$. Figure (3.7) shows that the two solutions are in excellent agreement.

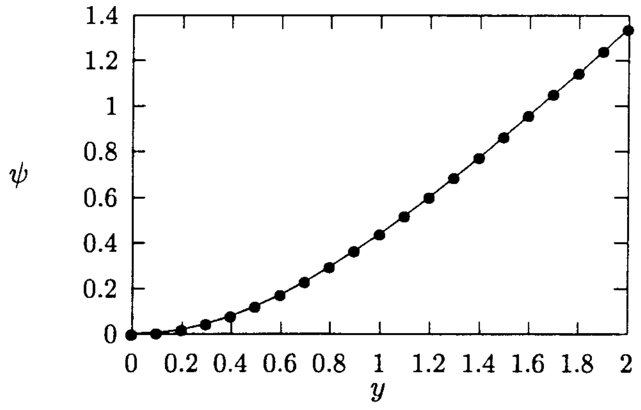


Figure 3.7: Comparison of exact solution (solid line) with numerical solution (solid circle). The parameters of the problem are $h = 2$,$\epsilon = 0$,$M_A = 0.5$ and $\psi_{wall} = 1.341529$.

The nonlinear nature of the governing equations for the parallel shear flow suggest that there might be other solutions than the one obtained above. This is actually easy to verify by looking at equation (3.59). Instead of stretching the y-coordinate axis with the density, it is possible to view the compressibility correction as an addition to some base incompressible result and hence write

$$\psi = \log\cosh(h\,y') + \log(\rho), \tag{3.62}$$

which, when substituted into equation (3.60), will give the following implicit analytic result for the density of the flow

$$\rho = \left[1 + \frac{\gamma - 1}{2} M_A^2 (1 - \frac{\tanh^2 h}{\rho^2})\right]^{1/(\gamma-1)} . \tag{3.63}$$

Note that these solutions have different values for the stream function on the wall and hence have different circulations.

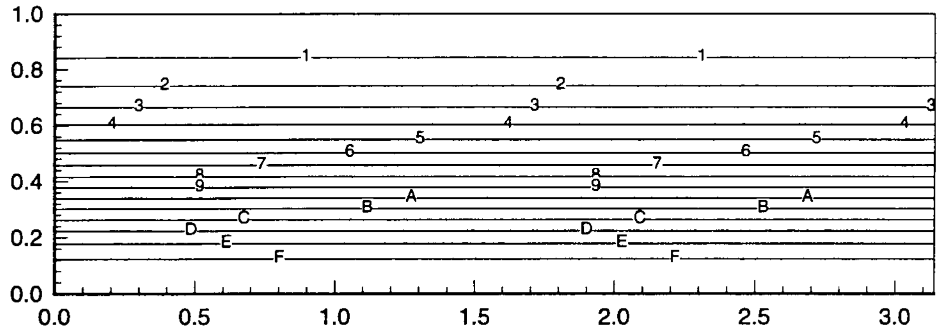


Figure 3.8: Vorticity contours for $h = 2$, $\epsilon = 0$, $\psi_{wall} = \log[\cosh h]$ and $M_A = 0$. The contours are equally spaced. Level 1 corresponds to a value of $\omega_1 = 0.128734$ and Level F corresponds to $\omega = 0.941895$.

3.6.3 Parameter search

Equation (3.58) suggests that for a fixed Mach number, the greatest drop in density will be realized for the vortices with the smallest cores. These vortices correspond to $\epsilon = 1$ and are the limit of an array of point vortices. The other limit corresponds to $\epsilon = 0$ which represents a parallel shear flow. Since we have explored the parallel shear flow limit analytically, we shall now choose two different values of $\epsilon = 0.5, 0.8$ to conduct our parameter search. The vorticity contours for each of these values as well as that of $\epsilon = 0$ is shown in Figures (3.8), (3.9) and (3.10). All the solutions depicted showed second order convergence consistant with the numerical discretization described in section 3.4.

Furthermore, we shall choose two values of the wall height, $h = 2, 6$. We have shown that $h = 6$ is a good approximation for $h \to \infty$, and by taking $h = 2$ we will study the effect of walls on the flow structure. In each of the

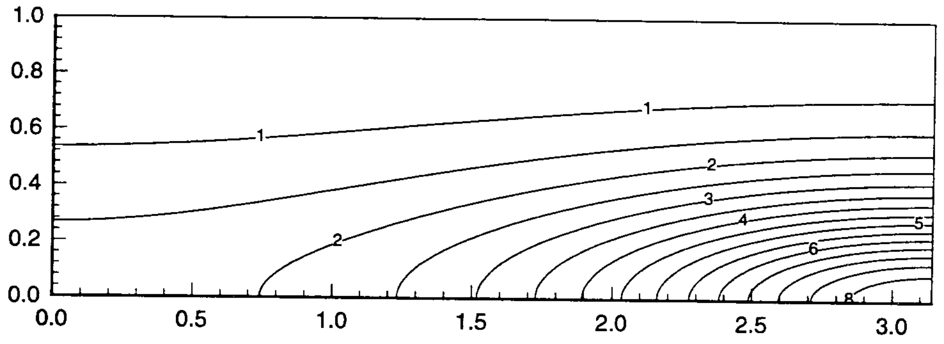


Figure 3.9: Vorticity contours for $h = 2$, $\epsilon = 0.5$, $\psi_{wall} = \log\cosh h$ and $M_A = 0$. The contours are equally spaced. Level 1 corresponds to a value of $\omega_1 = 0.220126$ and Level F corresponds to $\omega = 2.56006$.

above computations, the Mach number is set at $M_A = 0$ and is increased in steps of $\delta M_A = 0.01$. Test runs were done on various grid sizes $(N, M) = (81,81), (101,101), (121,121)$ and finally the computations were performed on a $(N, M) = (101, 101)$ grid. While increasing the Mach number of the flow, we noticed that the number of iterations for convergence to a solution increased as well. For all the cases tested above, the solutions would exhibit a drop in the core density with an increase in the Mach number. After some critical value of M_A, the density in the core would start increasing, and finally the solution would converge to a parallel shear flow regardless of the input parameters. The maximum attainable Mach number as well as the density drop in the core are tabulated in Table (3.3). Typical contours of the stream function and density are shown in Figures (3.11), (3.12), (3.13) and (3.14).

Even though the drop in the core density was much more significant for $\epsilon = 0.8$ than for $\epsilon = 0.5$, none of the above searches lead to local supersonic

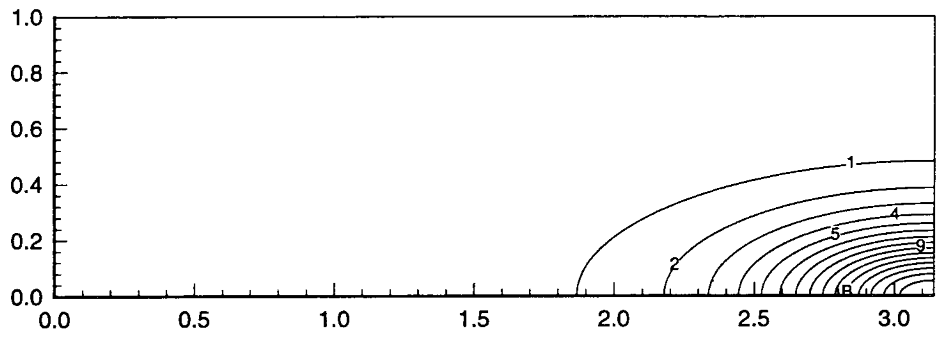


Figure 3.10: Vorticity contours for $h = 2$, $\epsilon = 0.8$, $\psi_{wall} = \log(\cosh h)$ and $M_A = 0$. The contours are equally spaced. Level 1 corresponds to a value of $\omega_1 = 0.647702$ and Level F corresponds to $\omega = 9.35945$.

flow anywhere in the computational domain. As is shown in Figure (3.15), the local Mach number in the flow increases from the specified value on the wall to a value slightly higher and decreases to zero at the core of the vortex. Solving equation (3.46) with a suitable iterative solver will increase the resolution of the problem, and this could possibly lead to finding supersonic flow.

3.6.4 Effect of wall height and compressibility

We shall attempt to define a vortex core so as to investigate the effects of compressibility and wall height on the structure of vortices with continuous vorticity distribution. Figure (3.16) depicts the horizontal velocity at the right- hand boundary, $u(x = \pi, y)$. Figure (3.17) depicts the vertical velocity profile on the symmetry line $y = 0$. We shall denote the distance from the peak of the horizontal velocity, $u_{max}(x = \pi, y)$, to the vortex core by b_1.

h	ϵ	M_A	$\rho(x = \pi, y = 0)$
2.00	0.50	0.27	0.9634
2.00	0.80	0.32	0.6365
6.00	0.50	0.175	0.9853
6.00	0.80	0.275	0.90833

Table 3.3: A list of the maximum density drop in the core for two different values of the wall height, h, and vorticity concentration, ϵ.

Similarly, we shall define the distance from the peak of the vertical velocity, $v_{max}(x, y = 0)$, to the core by b_2. In Figures (3.18) and (3.19), we set $\epsilon = 0.5$ and $M_A = 0.0$ and hold the circulation in the domain, $\Gamma = \pi$, fixed and compute the solutions for various wall heights. The curve plotted in Figure (3.18) flattens out at $h = 3$ and remains horizontal as $h \to 0$. This plot shows that the location of the maximum horizontal velocity peak used in Figure (3.16) to define b_1 approaches the channel wall as $h \to 3$ and remains there for all subsequent decreasing values of h. Figure (3.19) shows that the location of the maximum vertical velocity peak used to define b_2 in Figure (3.17) is tending towards zero as $h \to 0$. Hence, we can conclude that in the limit of decreasing wall height, the recirculation regions present in Stuart vortex solutions vanish. In Figure (3.20) we set $\epsilon = 0.5$ and $\psi_{wall} = \log[\cosh h]$, increase the Mach number M_A, and measure the circulation in the domain for various values of the channel height. This shows that for a given vorticity distribution in which the parameters are fixed, the circulation in the domain is greatest for smaller values of the wall height. Also, it further confirms that $h = 6$ is a good numerical approximation for $h \to \infty$ since

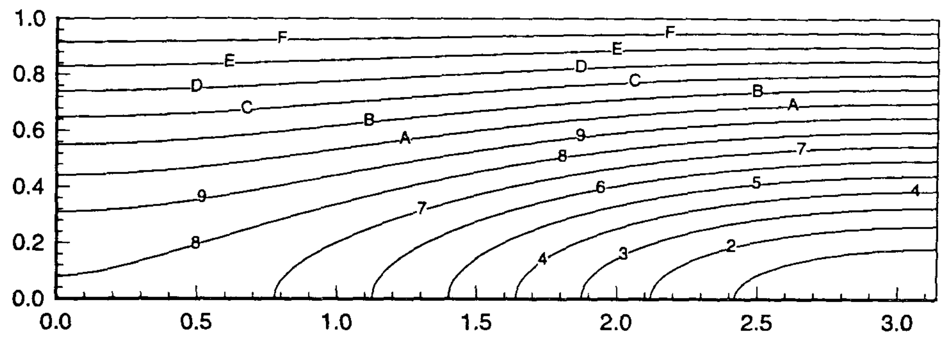


Figure 3.11: Equally spaced contours of the stream function for $h = 2$, $\epsilon = 0.8$, $\psi_{wall} = \log\cosh h$ and $M_A = 0.32$. Level 1 corresponds to a value of -1.43116, and Level F corresponds to 1.14126.

the difference between the values of the circulations plotted for various wall heights decreases rapidly such that the difference between $h = 4$ and $h = 6$ is hardly noticeable. It must be noted here that the computations for the various wall heights was stopped at the highest attainable Mach number.

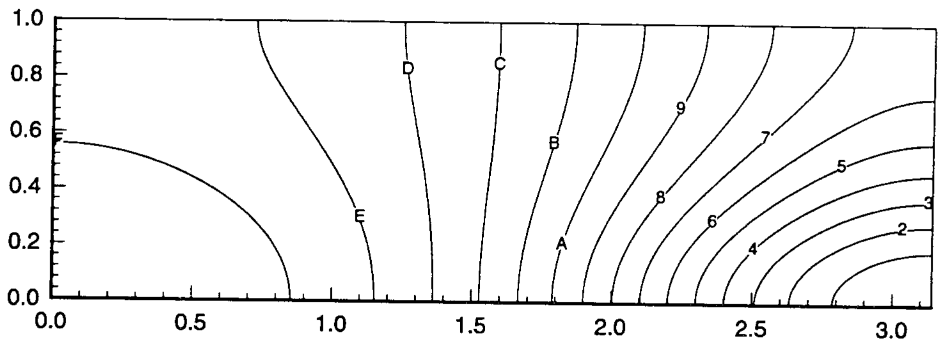


Figure 3.12: Equally spaced contours of the density for $h = 2$, $\epsilon = 0.8$, $\psi_{wall} = \log[\cosh h]$ and $M_A = 0.32$. Level 1 corresponds to 0.66083, and Level F corresponds to 1.00293.

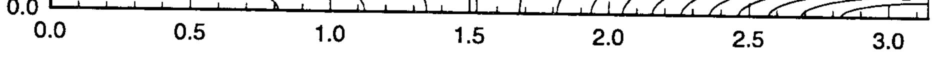


Figure 3.13: Equally spaced contours of the density for $h = 6$, $\epsilon = 0.8$, $\psi_{wall} = \log[\cosh h]$ and $M_A = 0.275$. Level 1 corresponds to 0.915557, and Level F corresponds to 1.01879.

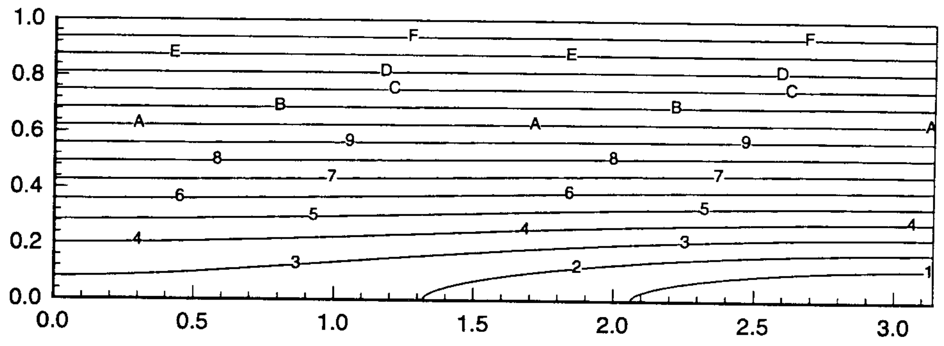


Figure 3.14: Equally spaced contours of the stream function for $h = 6$, $\epsilon = 0.8$, $\psi_{wall} = \log[\cosh h]$ and $M_A = 0.275$. Level 1 corresponds to a value of -0.79971 and Level F corresponds to 4.80415

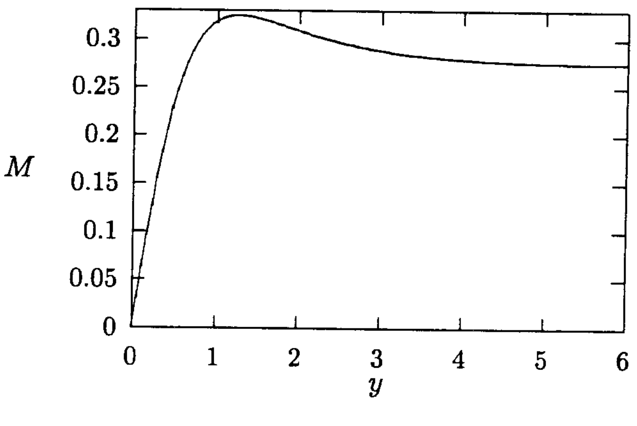


Figure 3.15: Local mach number profile at $x = \pi$. Parameters of the problem are $h = 6$, $\epsilon = 0.8$, $M_A = 0.275$ and $\psi_{wall} = \log[\cosh(h)]$.

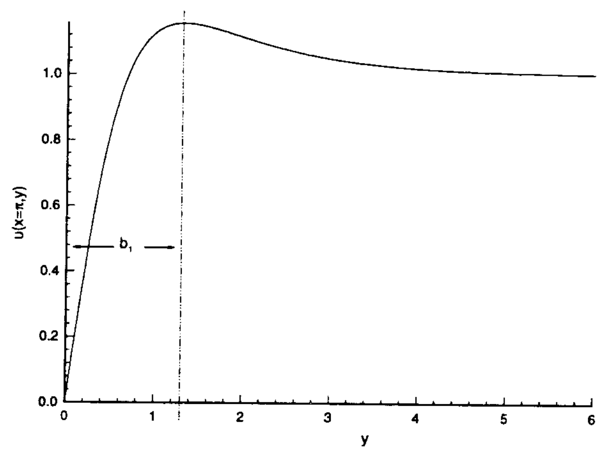


Figure 3.16: Horizontal velocity profile, $u(x = \pi, y)$, for unbounded Stuart vortex solution given by equation (3.5) in which $\epsilon = 0.5$.

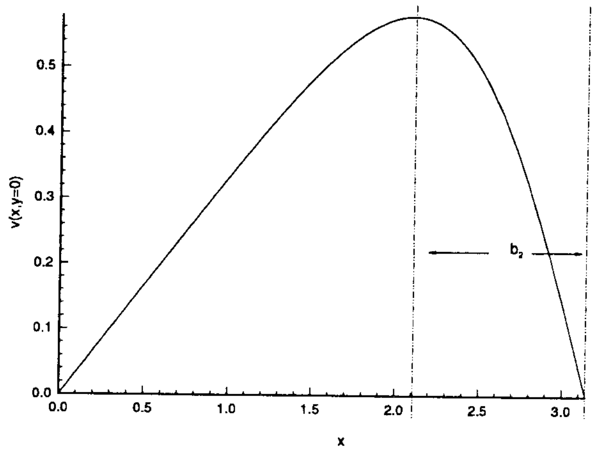


Figure 3.17: Vertical velocity profile, $v(x, y = 0)$, for unbounded Stuart vortex solution given by equation (3.5) in which $\epsilon = 0.5$.

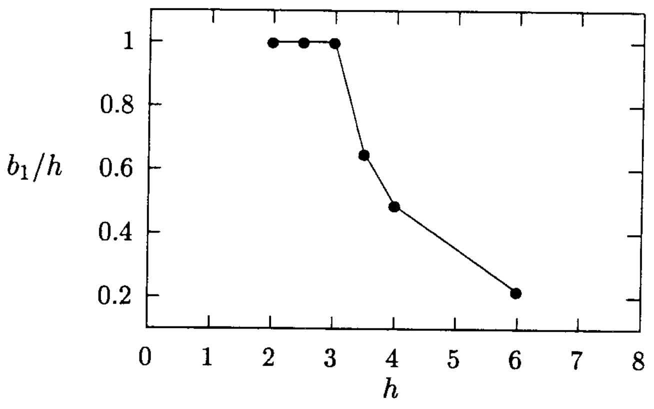


Figure 3.18: Horizontal extent of the defined vortex core plotted versus varying channel height. $\Gamma = \pi$, $M_A = 0$, and $\epsilon = 0.5$ are held fixed

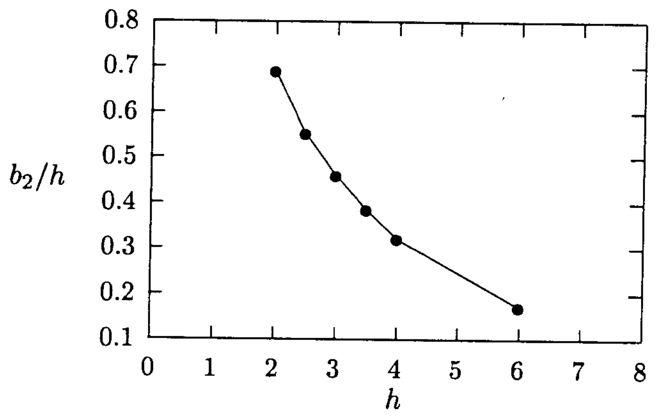


Figure 3.19: Vertical extent of the defined vortex core plotted versus varying channel height. $\Gamma = \pi$, $M_A = 0$, and $\epsilon = 0.5$ are held fixed

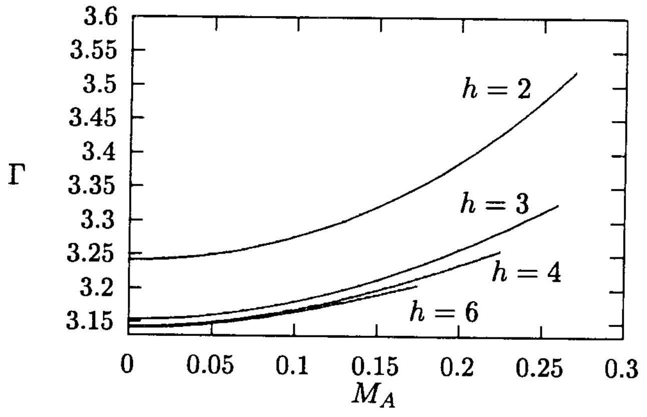


Figure 3.20: Circulation measured as a function of Mach number for various wall heights

Chapter 4

Conclusion

In chapter 2 we have shown that it is possible to have steady flow of an infinite, periodic array of vortices in a compressible flow and, furthermore, that this array can exhibit transonic shock-free flow. It has also been illustrated that by taking special care in treating the singularity of the transformation at infinity, it is possible to reduce a compressible transonic flow to a strictly linear problem in the hodograph plane. The evacuated vortex core results indicate that such solutions can be constructed in the hodograph plane, but that they all exhibit cuspidal behaviour which indicates the onset of limit lines.

Further extensions of this work include investigation of the stability of this vortex array. We have shown that steady solutions exist and now we must determine whether these solutions are stable. Also of interest is to determine whether there exists a class of compressible free boundary problems that lead to a linear system of equations in the hodograph plane. If so, this would then allow for the use of the principle of superposition in constructing compressible flow streamlines for a variety of flows in the hodograph plane akin to potential flow theory.

Note that the vorticity in the hollow vortex core problem of chapter 2 was concentrated into vortex sheets on the surfaces of the vortices. In chapter 3, by extending the Stuart vortex solution to include compressibility, we have constructed two dimensional steady compressible vortex flows with continuous distribution of vorticity. Increasing the channel width, $h \to \infty$, shows that compressible Stuart-type vortices in an unbounded domain exist. Furthermore, it suggests that the circulation is a specific function of the Mach number. Hence, in order to solve for the unbounded vortex flow, it is necessary to continue the generalized Stuart vortex solution given in section 3.1. By taking

$$\psi = A\,y, \qquad y \to \infty$$

we inflate the system of equations given by equation (3.46) by one unknown, A, and use as the boundary condition at infinity a chosen value for the circulation of the problem, Γ. The specified value of the circulation then will determine the value of A uniquely. This gives an integral constraint for the boundary condition at infinity that can easily be dealt with through the use of bordered matrices.

We have shown that the effect of decreasing the channel width while fixing the circulation in the problem is to wash out the recirculating regions and that the limit of very small channel height is inevitably a parallel shear flow. Furthermore, exact analytical solutions for the special case of the compressible parallel shear flow are given in implicit form. Even though no local supersonic regions where found for the Stuart-type vortices, it is believed that increasing the resolution will most probably help in increasing the Mach numbers achieved in this work. The fact that the number of iterations per solution increased as the Mach number was raised suggests that if an iterative matrix solver with an appropriate pre-conditioner is chosen, perhaps locally supersonic flow can be achieved.

Constructing the compressible homentropic Stuart vortices in an unbounded domain is certainly a natural extension of this work. These solutions can be very valuable to studies of shock vortex interaction in compressible media as well as studies in turbulence and aero acoustics. The extension of this work to the other limit of homenthalpic flow, as described in section (3.2), can also be very enlightening since allowing the entropy to vary may result in regions of steep entropy gradients in the flow. Certainly a stability analysis of these flows is also necessary to make the investigation of the compressible Stuart-type vortices complete.

Another interesting thought that has risen from this work is the possibility of extending results in potential flow to include continuous vorticity distributions. The Stuart vortex solution can be viewed as extending an infinite array of point vortices. Mallier and Maslowe [15] capitalize on this idea by extending the potential for a row of counter rotating vortices. Since potential flow theory has been used to a great extent by scientists and engineers to model flows (i.e., wake behind a cylinder, flow over an airfoil, etc.), this extension to include continuous vorticity distributions would make the models more realistic.

Appendix A

Asymptotic analysis of Chaplygin's equation

Introducing the following local coordinates in the hodograph plane,

$$\alpha = q - a, \qquad \beta = \theta - \pi, \tag{A.1}$$

and substituting this result into equation (2.10) leads to

$$\begin{aligned}\frac{\partial^2\psi}{\partial\alpha^2} + (1 - M_\infty^2)\frac{\partial^2\psi}{\partial\beta^2} = &- (\alpha^2 + 2\alpha)(1 - p_1(\alpha+1)^2)\frac{\partial^2\psi}{\partial\alpha^2} \\ &- (\alpha+1)(p_2 - p_3(\alpha+1)^2)\frac{\partial\psi}{\partial\alpha} \\ &+ p_4(\alpha^2 + 2\alpha)\frac{\partial^2\psi}{\partial\beta^2},\end{aligned} \tag{A.2}$$

where $p_1 = \frac{\gamma-1}{2}M_\infty^2$, $p_2 = 1 + p_1$, $p_3 = \frac{\gamma-3}{2}M_\infty^2$ and $p_4 = \frac{\gamma+1}{2}M_\infty^2$.

Near the singularity at (a, π), (A.2) takes the form

$$\frac{\partial^2 \psi}{\partial \alpha^2} + (1 - M_\infty^2)\frac{\partial^2 \psi}{\partial \beta^2} = 0 \tag{A.3}$$

along with the corresponding boundary condition,

$$\frac{\partial \psi}{\partial \beta} = 0, \qquad \beta = 0. \tag{A.4}$$

We introduce local polar coordinates (s, δ) defined by

$$se^{\imath\delta} = (q - a) + \imath\frac{\theta - \pi}{\sqrt{1 - M_\infty^2}}. \tag{A.5}$$

Then, all possible solutions of the above equation are of the form

$$\psi = s^{2m} \cos 2m\delta \tag{A.6}$$

where m has integral values, as well as

$$\psi = \log(s) + C \tag{A.7}$$

where C is a constant. We choose the type of the singularity in the hodograph plane as required by the formulation of the problem in the physical plane (as discussed in section 2.2.4). Thus we write $\psi = \psi_0 + \psi_1$ where

$$\psi_0 = \log(s) \tag{A.8}$$

so that, on identifying the dominant terms on the right-hand side of (A.2),

$$\frac{\partial^2 \psi_1}{\partial s^2} + \frac{1}{s}\frac{\partial \psi_1}{\partial s} + \frac{1}{s^2}\frac{\partial^2 \psi_1}{\partial \delta^2} = -2p_5 s \cos\delta \cos\frac{2\delta}{s^2} - (p_2 - p_3)\cos\frac{\delta}{s}, \qquad \text{(A.9)}$$

where $p_5 = 1 - p_1 + \frac{p_4}{1-M_\infty^2}$. Solving the above equation leads to the complementary function

$$\psi_1 = \frac{p_5}{8} s \cos 3\delta - \frac{p_6}{2} s \log s \cos\delta. \qquad \text{(A.10)}$$

It is now evident that the $\log s$ singularity has higher order corrections that all go to zero smoothly as $(s, \delta) = (0, 0)$. Since we intend to subtract the singularity from equation (2.10) and solve for the regular corrections, the exact form of the singularity is crucial. Any other form of the singularity, upon subtraction from equation (2.10), would lead to residual singularities in the modified equation.

Appendix B

Estimation of ψ_v using matched asymptotics

In this Appendix we find a leading order approximation to the value of the stream function at the boundary of the evacuated vortex. The incompressible solution [12] for a row of point vortices is given by

$$\psi(x,y) = -\frac{\Gamma}{4\pi}\log\left[\frac{1}{2}\left(\cosh(2\pi y/L) - \cos(2\pi x/L)\right)\right]. \qquad \text{(B.1)}$$

We opt to use polar coordinates and hence use the transformation

$$y = r\sin\theta,$$

$$x = r\cos\theta,$$

to get the incompressible solution $\psi = \psi(r, \theta)$. Furthermore, we take the limit of (B.1) as $r \to 0$ and find the leading order behaviour to be given by

$$\lim_{r \to 0} \psi(r, \theta) \sim \frac{\Gamma}{2\pi} \left[-\log(\pi/L) - \log(r) \right]. \tag{B.2}$$

The solution for an isolated hollow vortex [18] in a compressible medium is given by

$$\psi(s) = \frac{\Gamma}{4\pi} \left[(1-s)^{\gamma-1} \log s + \frac{1}{\gamma - 1} \int_1^s \log \xi (1-\xi)^{\frac{2-\gamma}{\gamma-1}} \, d\xi \right] + \psi_v, \tag{B.3}$$

where $s = R^2/r^2$ and $R = \frac{\Gamma(\gamma-1)^{\frac{1}{2}}}{2\sqrt{2}\pi c_o}$ where the subscript o denotes stagnation conditions. We shall define

$$F(s; \gamma) = \frac{1}{\gamma - 1} \int_0^s \log \xi \, (1-\xi)^{\frac{2-\gamma}{\gamma-1}} \, d\xi$$

and hence find the leading order behaviour of $\psi(s)$ as $s \to 0$ as follows:

$$\psi(s) \sim \frac{\Gamma}{2\pi} \left[-\log r + \log R - \frac{1}{2} F(1; \gamma) \right] + \psi_v \tag{B.4}$$

The incompressible solution, (B.2), is an outer solution which is matched to the solution for an isolated hollow vortex, (B.4), giving an approximation for the value of the stream function on the boundary of the evacuated vortex.

$$\psi_v = \frac{\Gamma}{4\pi} \left[F(1; \gamma) - \log a^2 \right]. \tag{B.5}$$

Appendix C

Jacobian entries for Stuart vortex problem

Section (3.4) discusses how to obtain the Jacobian of the system of equations (3.44) and (3.45). The linearized form of equation (3.44) is written as

$$A\delta\psi_{xx}+B\delta\psi_{yy}+C\delta\psi_x+D\delta\psi_y+E\delta\psi+F\delta\rho+G\delta\rho_x+H\delta\rho_y+\mathcal{L}=0. \quad \text{(C.1)}$$

We solve for the density correction, $\delta\rho$, from equation (3.45),

$$\begin{aligned}
\delta\rho &= \frac{-1}{Q}\left(R\,\delta\psi_x+S\,\delta\psi_y+T\,\delta\psi+\mathcal{G}_1(\psi_0,\rho_0)\right),\\
R &= \frac{(\gamma-1)M_A^2}{\rho_0^2}\psi_{0x},\\
S &= \frac{(\gamma-1)M_A^2}{h^2\rho_0^2}\psi_{0_{y'}},\\
T &= -(\gamma-1)M_A^2(1-\epsilon^2)\exp(-2\psi),\\
Q &= (\gamma-1)\rho_0^{\gamma-2}-\frac{\gamma-1}{\rho_0^3}M_A^2\left(\psi_{0_x}^2+\frac{1}{h^2}\psi_{0_{y'}}^2\right), \qquad \text{(C.2)}
\end{aligned}$$

where $\mathcal{G}_1(\psi_0, \rho_0)$ is equation (3.45) evaluated at the latest guess and h is the position of the wall. We can now substitute the above result into equation (C.1). This leads to the following system:

$$\begin{aligned} cp2dx\,\delta\psi_{xx} &+ cp2dy\,\delta\psi_{yy} + cp1dx\,\delta\psi_x \\ &+ cp1dy\,\delta\psi_y + cpmd\,\delta\psi_{xy} + cp\,\delta\psi + ce = 0. \end{aligned} \tag{C.3}$$

The entries of the Jacobian matrix $\mathbf{A}_{i,j}$, and the right-hand side matrix, $\mathbf{B}_{i,j}$, is given by the following:

$$\begin{aligned} a_{i+1,j} &= cp2dx/\Delta x^2 + cp1dx/2\Delta x, \\ a_{i-1,j} &= cp2dx/\Delta x^2 - cp1dx/2\Delta x, \\ a_{i,j} &= -2\ cp2dx/\Delta x^2 - 2\ cp2dy/\Delta y^2 + cp, \\ a_{i,j+1} &= cp2dy/\Delta y^2 + cp1dy/2\Delta y, \\ a_{i,j-1} &= cp2dy/\Delta y^2 - cp1dy/2\Delta y, \\ a_{i+1,j+1} &= cpmd/4\ \Delta x\,\Delta y, \\ a_{i-1,j-1} &= cpmd/4\ \Delta x\,\Delta y, \\ a_{i+1,j-1} &= cpmd/4\ \Delta x\,\Delta y, \\ a_{i-1,j+1} &= cpmd/4\ \Delta x\,\Delta y, \end{aligned} \tag{C.4}$$

where Δx and Δy are defined in section 3.4 and

$$\begin{aligned} cp2dx &= A - (GR/Q), \\ cp2dy &= B - (HS/Q), \\ cp1dx &= C - (FR/Q) + G[(RQ_x/Q^2) - (R_x/Q) - (T/Q)] \end{aligned}$$

$$\begin{aligned}
&+ \quad H[(Q_y R/Q^2) - (R_y/Q)], \\
cp1dy &= D - (FS/Q) + G[(SQ_x/Q^2) - (S_x/Q)] \\
&+ \quad H[(Q_y S/Q^2) - (S_y/Q) - (T/Q)], \\
cp &= E - (FT/Q) + G[(Q_x T/Q^2) - (T_x/Q)] + H[(TQ_y/Q^2) - (T_y/Q)], \\
cpmd &= -(SG/Q) - (HR/Q), \\
ce &= \mathcal{L} - (F\mathcal{G}/Q) + G[(Q_x\mathcal{G}/Q^2) - (\mathcal{G}_x/Q)] \\
&+ \quad H[(Q_y\mathcal{G}/Q^2) - (\mathcal{G}_y/Q)].
\end{aligned} \tag{C.5}$$

The functions Q, R, S, T and $\mathcal{G}$ have already been defined above. The remaining functions from the Jacobian entries above are related to the initial guess through the following relationships:

$$\begin{aligned}
A &= 1/\rho_0 \\
B &= 1/(h^2\,\rho_0) \\
C &= -\rho_{0_x}/\rho_0^2 \\
D &= -\rho_{0_x}/(h^2\,\rho_0^2) \\
E &= 2\,\rho_0(1-\epsilon^2)\exp(-2\,\psi_0) \\
F &= -\psi_{0_{xx}}/\rho_0^2 + 2\rho_{0_x}\psi_{0_x}/\rho_0^3 - \psi_{0_{yy}}/(h^2\,\rho_0^2) \\
&+ \quad 2\,\rho_{0_y}\psi_{0_y}/(h^2\,\rho_0^3) - (1-\epsilon^2)\exp(-2\,\psi_0) \\
G &= -\psi_{0_x}/\rho_0^2 \\
H &= -\psi_{0_y}/(h^2\,\rho_{0_y}) \\
\mathcal{L} &= (\psi_{0_x}/\rho_0)_x + (\psi_{0_y}/\rho_0)_y - \rho_0(1-\epsilon^2)\exp(-2\,\psi_0)
\end{aligned} \tag{C.6}$$

The right-hand side matrix is given by the following everywhere in the interior of the grid,

$$b_{i,j} = -ce, \tag{C.7}$$

and by the boundary conditions listed in section (3.4) on the boundaries.

Appendix D

Fortran listing for hollow core vortex

```
*****
***  Using the capacitance matrix approach
***  to solve the second order finite difference
***  discretization of the problem.  The point at
***  infinity is treated specially (dtpsi=0 used directly)
***  Don't forget to change dimensions for the mapping !!!
***  Need to change in subroutine map as well as function psin
*****
      real*8 q,tht,ainf,minf,bta,g,a,b,d,e,arg3,arg4,arg5
      real*8 dq,dtht,pi,t1,t2,tm1,mv,diff,exact,extra,pert
      real*8 force,minf2,ainf2,ext,arg1,arg2,filter
      real*8 term,bndq0,bndq1,bndtop,botconst,topconst,capconst
      integer n,m,i,j,il,iu,ia,is,js,ias
```

```
 integer j1,j2,j3,j4,j5,nm,ik,ip,kmap,nn
 parameter(n=1001,m=256,il=1,iu=n,nm=n*(m+1),ip=1,kmap=10)
 parameter(ainf=0.2d0,minf=0.d0,g=1.4d0,nn=((n-1)/kmap)+1)
 double precision aa(n),bb(n),dd(n),cc(n),res6(nm,ip)
 double precision psik(n,m+1),y(m+1),yz(nm),psi(n,m+1)
 double precision uu(nm),vv(nm),res(ip,ip),rhs(nm)
 double precision unit(ip,ip),res2(ip,ip),indx(ip),yy(ip,ip)
 double precision res3(ip,ip),res4(ip,ip),res5(nm,ip)
 double precision ut(n),vt(n),wk(n),xt(nn,m+1),yt(nn,m+1)
 open(67,file='kay.out')
 open(70,file='stag.out')
 open(71,file='top.out')
 open(73,file='inf.out')
 open(74,file='side.out')
 write(*,*) 'parameters of the problem'
 pi = dacos(-1.d0)
 ainf2 = ainf*ainf
 minf2 = minf*minf
 bta = (minf2/(2.d0+((g-1)*minf2)))/(ainf2)
 write(*,*) 'bta= ',bta
 dq = 1.d0/(n-1)
 dtht = (pi)/(2.d0*(m))
 tm1 = (ainf2-1.d0)*(g-1.d0)*minf2+2.d0*ainf2
 mv = dsqrt(2.d0*minf2/tm1)
 write(*,*) 'dq2= ',dq*dq
 write(*,*) 'dtht2= ',dtht*dtht
```

```
      write(*,*) 'minf= ',minf
      write(*,*) 'ainf= ',ainf
      write(*,*) 'mmortex= ',mv
      write(*,*)
      write(*,*) 'checking points for the computation'

*****
***   initializing the unit matrix for the capacitance approach
*****
      do 123  i=1,ip
         do 124  j=1,ip
            unit(i,j) = 1.d0
 124     continue
 123  continue

*****
***  setting up the correction matrices
*****
      is = (ainf/dq)+1
      js = m+1
      ias = n*(js-1)+is
      ia = ias

      do 55  i=1,nm
         uu(i)=0.d0
55    continue
```

```
      uu(ia)=1.d0
      do 66  i=1,nm
         vv(i)=0.d0
 66   continue
      j=m+1
         q=(is-1)*dq
         write(*,*) 'at infinity q= ',q
         if(q.ne.ainf)  pause 'something is wrong'
         tht = (pi/2.d0)+(j-1)*dtht
         j1 = n*(j-1)+is+1
         j2 = n*(j-1)+is
         j3 = n*(j-1)+is-1
         j4 = n*(j-2)+is
         j5 = n*(j-3)+is
         vv(j1)= -a(g,bta,dq,q)
         vv(j2)= 3.d0-d(g,bta,dq,dtht,q)
         vv(j3)= -b(g,bta,dq,q)
         vv(j4)= -4.d0-2.d0*e(g,bta,dtht,q)
         vv(j5) = 1.d0

*****
***   Re-ordering and transforming the U matrix
*****


      do 51  i=1,n
         do 100  j=1,m+1
```

```
             ia = n*(j-1)+i
             y(j) = uu(ia)
 100      continue

          call cosft1 (y,m)

          do 111  j=1,m+1
             ia = n*(j-1)+i
             uu(ia) = y(j)
 111      continue
 51    continue

*****
***   setting up the equations for each mode
*****

       do 5  j=1,m+1
*****
***  constructing the "i+1" diagnol
*****
       do 15  i=2,n-1
          q=(i-1)*dq
          t1 = q*q*(1.d0-(g-1.d0)*bta*q*q)/(dq*dq)
          t2 = q*(1.d0-(g-3.d0)*bta*q*q)/(2.d0*dq)
       aa(i) = t1+t2
 15    continue
```

```
***  boundary conditions
      aa(il) = 4.d0
      aa(iu) = 0.d0

*****
***  constructing the "i" diagnol
*****

      do 20  i=2,n-1
      q=(i-1)*dq
      tht = (pi/2.d0)+(j-1)*dtht
      t1 = -2.d0*q*q*(1.d0-(g-1.d0)*bta*q*q)/(dq*dq)
      filter = 8.d0*(cos((j-1)*dtht)-1.d0)/(dtht*dtht)
*      filter = -4.d0*(j-1)*(j-1)
      t2 = filter*(1.d0-(g+1)*bta*q*q)
      dd(i) = t1+t2
 20   continue

***  boundary conditions
      dd(il) = -3.d0
      dd(iu) = 1.d0

*****
*** constructing the "i-1" diagnol
*****
```

```
      do 25  i=2,n-1
         q=(i-1)*dq
         t1 = q*q*(1-(g-1)*bta*q*q)/(dq*dq)
         t2 = -q*(1-(g-3)*bta*q*q)/(2.d0*dq)
      bb(i) = t1+t2
 25   continue

***  boundary conditions
      bb(il) = 0.d0
      bb(iu) = 0.d0

*****
***  solving the first of the P-auxilary problems
*****

      do 225  i=1,n
         ik = n*(j-1)+i
         cc(i) = uu(ik)
 225  continue

*****
***  using the Sherman-Morrison formula for the
***  tri-diagnol solves ... see diagram
*****
```

```
      do 1022 i=1,n
         ut(i) = 0.d0
         vt(i) = 0.d0
 1022 continue
      ut(il) = 1.d0
      vt(3) = -1.d0

      call sy(il,iu,bb,dd,aa,cc)
      do 1011  i=1,n
         wk(i) = cc(i)
         cc(i) = ut(i)
 1011 continue

      call sy(il,iu,bb,dd,aa,cc)

      call dotmat(vt,cc,1,1,n,res)
      botconst = 1.d0 + res(1,1)
      call dotmat(vt,wk,1,1,n,res)
      topconst = res(1,1)
      capconst = -1.d0*topconst/botconst
      do 1012  i=1,n
         cc(i) = capconst*cc(i)
 1012 continue
      call addmat(wk,cc,n,1,cc)

****  sherman-morrison formula done
```

```
*******

      do 226  i=1,n
         ik=n*(j-1)+i
         uu(ik)=cc(i)
 226  continue
 5    continue
      write(*,*) 'it is working for auxilary problem'

*****
***   taking the result of the auxilary problem and transforming
***   it back to real space
*****

      do 83  i=1,n
         do 84  j=1,m+1
            ia = n*(j-1)+i
            y(j) = uu(ia)
 84      continue

         call cosft1(y,m)

         do 85  j=1,m+1
            ia = n*(j-1)+i
            uu(ia) = (2.d0/m)*y(j)
 85      continue
 83      continue
         write(*,*) 'I have z in the real space'

*****
***  Performing the PxP matrix inversion
*****


         call dotmat(vv,uu,ip,ip,nm,res)
         write(*,*) 'the product of vv and z= ',res(1,1)
         call addmat(res,unit,ip,ip,res2)
         write(*,*) 'the addition by one gives= ',res2(1,1)
         call matinv(res2,ip,ip,indx,yy)
         write(*,*) 'the inversion routine gives= ',yy(1,1)
         ext = 1/yy(1,1)
         write(*,*) 'this is inversion by hand= ',ext
******
*****
***   Constructing the Forcing Term
*****
******
***  the interior points


      do 2  i=2,n-1
         do 3 j=1,m+1
      q = (i-1)*dq
      tht = (pi/2.d0) + (j-1)*dtht
```

```
         ia = n*(j-1)+i
         if(q.eq.ainf.and.tht.eq.pi) then
           rhs(ia) = 0.d0
         else
           rhs(ia) = force(g,ainf,minf,q,tht)
*            rhs(ia) = 0.d0
         endif
 3       continue
 2       continue


***   the right hand side derivative condition


        i=1
        do 444  j=1,m+1
           q = (i-1)*dq
           tht = (pi/2.d0) + (j-1)*dtht
           ia = n*(j-1)+i
           rhs(ia) = 2.d0*dq*bndq0(ainf,minf,tht)
*           rhs(ia) = 0.d0
 444    continue


***   the fixed value of psi


        i=n
```

```
      do 4  j=1,m+1
         tht = (pi/2.d0) + (j-1)*dtht
         ia = n*(j-1)+i
         rhs(ia) = bndq1(ainf,minf,tht)
*         rhs(ia) = cos(2.d0*tht)
 4       continue

***  the bottom boundary ... dtpsi=f(q)

      j=1
      do 114  i=2,n-1
         q=(i-1)*dq
         tht = (pi/2.d0)+(j-1)*dtht
         ia = n*(j-1)+i
        term = 2.d0*dtht*e(g,bta,dtht,q)*bndtop(ainf,minf,q)
*        rhs(ia)=force(g,ainf,minf,q,tht)+term
*        rhs(ia) = 0.d0
 114  continue
        write(*,*) 'forcing function complete'


*****
***   transforming the rhs vector to Fourier Space
*****
      do 512  i=1,n
         do 200  j=1,m+1
```

```
            ia = n*(j-1)+i
            y(j) = rhs(ia)
  200     continue


          call cosft1 (y,m)


          do 121  j=1,m+1
            ia = n*(j-1)+i
            rhs(ia) = y(j)
  121     continue
  512   continue


*****
***   setting up the equations for each mode
*****


        do 151  j=1,m+1
*****
***  constructing the "i+1" diagnol
*****
        do 115  i=2,n-1
           q=(i-1)*dq
           t1 = q*q*(1-(g-1)*bta*q*q)/(dq*dq)
           t2 = q*(1-(g-3)*bta*q*q)/(2.d0*dq)
        aa(i) = t1+t2
 115   continue
```

```
***  boundary conditions
      aa(il) = 4.d0
      aa(iu) = 0.d0

*****
***  constructing the "i" diagnol
*****

      do 120  i=2,n-1
      q=(i-1)*dq
      tht = (pi/2.d0)+(j-1)*dtht
      t1 = -2.d0*q*q*(1-(g-1)*bta*q*q)/(dq*dq)
      filter = 8.d0*(cos((j-1)*dtht)-1.d0)/(dtht*dtht)
*      filter = -4.d0*(j-1)*(j-1)
      t2 = filter*(1.d0-(g+1)*bta*q*q)
      dd(i) = t1+t2
 120  continue

***  boundary conditions
      dd(il) = -3.d0
      dd(iu) = 1.d0

*****
*** constructing the "i-1" diagnol
*****
```

```
      do 125  i=2,n-1
         q=(i-1)*dq
         t1 = q*q*(1-(g-1)*bta*q*q)/(dq*dq)
         t2 = -q*(1-(g-3)*bta*q*q)/(2.d0*dq)
      bb(i) = t1+t2
 125   continue

***  boundary conditions
      bb(il) = 0.d0
      bb(iu) = 0.d0

*****
***   Solving the last tridiagnol problem
*****

      do 77  i=1,n
         ik = n*(j-1)+i
         cc(i)=rhs(ik)
 77   continue

*****
***  using the Sherman-Morrison formula for the
***  tri-diagnol solves ... see diagram
*****
```

```
      do 1042 i=1,n
         ut(i) = 0.d0
         vt(i) = 0.d0
 1042 continue
      ut(il) = 1.d0
      vt(3) = -1.d0

      call sy(il,iu,bb,dd,aa,cc)
      do 1031  i=1,n
         wk(i) = cc(i)
         cc(i) = ut(i)
 1031 continue
      call sy(il,iu,bb,dd,aa,cc)

      call dotmat(vt,cc,1,1,n,res)
      botconst = 1.d0 + res(1,1)
      call dotmat(vt,wk,1,1,n,res)
      topconst = res(1,1)
      capconst = -1.d0*topconst/botconst
      do 1032  i=1,n
         cc(i) = capconst*cc(i)
 1032 continue
      call addmat(wk,cc,n,1,cc)

****  sherman-morrison formula done
*******
```

```
      do 30  i=1,n
         psik(i,j) = cc(i)
 30   continue
 151  continue

*****
***   re-storing solution to physical plane
*****

      do 35  i=1,n
         do 40  j=1,m+1
           y(j) = psik(i,j)
 40      continue

      call cosft1(y,m)

      do 45  j=1,m+1
      psi(i,j) = (2.d0/m)*y(j)
 45   continue
 35   continue

      do 81  i=1,n
         do 82  j=1,m+1
            ia = n*(j-1)+i
```

```
          yz(ia)=psi(i,j)
 82     continue
 81   continue


*****
***   the final stage of the capacitance matrix approach
*****


      call dotmat(vv,yz,ip,ip,nm,res3)
      call dotmat(yy,res3,ip,ip,ip,res4)
      do 71  i=1,ip
         do 72  j=1,ip
            res4(i,j) = -res4(i,j)
 72      continue
 71      continue
      call dotmat(uu,res4,nm,ip,ip,res5)
      call addmat(yz,res5,nm,ip,res6)
      write(*,*) 'the capacitance matrix approach is DONE'

      do 181  i=1,n
         do 182  j=1,m+1
            ia = n*(j-1)+i
            psi(i,j) = res6(ia,1)
182      continue
181   continue
```

```
        i = (ainf/dq)+1
        do 88  j=1,m+1
           tht = (pi/2.d0) + (j-1)*dtht
           write(73,*) tht,psi(i,j)
 88     continue

        do 33  i=1,n
          do 34   j=1,m+1
             q = (i-1)*dq
             tht = (pi/2.d0)+(j-1)*dtht
             ia = n*(j-1)+i
             arg1 = (q-ainf)*(q-ainf)
             arg2 = ainf2*(tht-pi)*(tht-pi)/(1.d0-minf2)
             arg3 = -1.d0*ainf2*(tht-pi)*(tht-pi)
             arg4 = (1.d0-minf2)*(q-ainf)*(q-ainf)
             arg5 = ainf2*pi*pi/4.d0
             extra = arg3/(arg4+arg5)
             if(q.eq.ainf.and.tht.eq.pi) go to 34
             psi(i,j) = res6(ia,1)+log(arg1+arg2)+extra
*             psi(i,j) = res6(ia,1)+log(arg1+arg2)
 34       continue
 33     continue


*****
```

```
***  checking to see whether solution closes the
***  hodograph plane.
*****
       call closure(n,m+1,ainf,minf,g,dq,dtht,psi)


*****
***  checking for limit lines
*****
       call cusp(psi,n,m,dq,dtht,ainf,minf,g)
       write(*,*) 'No limit lines have appeared !!!'
*****
***  formatting output
*****
       if(minf.gt.0.1d0) go to 901

       i = (ainf/dq)+10
       do 89  j=1,m+1
           q = (i-1)*dq
           tht = (pi/2.d0)+(j-1)*dtht
           exact = pert(ainf,minf,q,tht)
           diff = exact - psi(i,j)
           write(74,701) tht,psi(i,j),exact,diff
 89     continue

       i=1
       do 54  j=1,m+1
```

```
           q = (i-1)*dq
           tht = (pi/2.d0)+(j-1)*dtht
           exact = pert(ainf,minf,q,tht)
           diff = exact - psi(1,j)
           write(70,701) tht,psi(1,j),exact,diff
54      continue
701     format(1x,4(f12.6,3x))

        j=1
        do 65  i=1,n
           q = (i-1)*dq
           tht = (pi/2.d0)+(j-1)*dtht
           exact = pert(ainf,minf,q,tht)
           diff = psi(i,1) - exact
           write(67,301) q,psi(i,1),exact,diff
65      continue
301     format(1x,4(f12.6,2x))

        j=m
        do 665  i=1,n
           q = (i-1)*dq
           tht = (pi/2.d0)+(j-1)*dtht
           exact = pert(ainf,minf,q,tht)
           diff = psi(i,m) - exact
           write(71,301) q,psi(i,m),exact,diff
665     continue
```

```
*****
***  output formatted for use with Tecplot
*****

 901   write(*,*) 'preparing output for tecplot'
      call map(g,ainf,minf,psi,n,m+1,xt,yt,nn,kmap)
      stop
      end

      subroutine addmat(a,b,n,m,res)
      implicit double precision (a-h,o-z)
      integer i,j,n,m
      double precision a(n,m),b(n,m),res(n,m)
*      write(*,*) 'inside matrix addition routine'
      do 10  i=1,n
         do 12  j=1,m
            res(i,j) = a(i,j)+b(i,j)
*     write(*,*) i,j,a(i,j),b(i,j),res(i,j)
 12      continue
 10   continue
*      write(*,*) 'outside matrix addition routine'
      return
      end
      subroutine dotmat(a,b,n,m,ijcmn,res)
      real*8 sum
      integer i,j,n,m,ij,ijcmn
      double precision a(n,ijcmn),b(ijcmn,m),res(n,m)

      do 10  i=1,n
      do 15  j=1,m
         sum = 0.d0
         do 20  ij=1,ijcmn
      sum = sum + a(i,ij)*b(ij,j)
20    continue
      res(i,j) = sum
15    continue
10    continue
      return
      end

      function a(g,bta,dq,q)
      real*8 g,bta,a,q,dq
      real*8 t1,t2
      a = (t1(g,bta,q)/(dq*dq)) + (t2(g,bta,q)/(2.d0*dq))
      return
      end

      function d(g,bta,dq,dtht,q)
      real*8 d,q,dtht,dq
```

```
real*8 t1,t3,g,bta
d = -2.d0*((t1(g,bta,q)/(dq*dq))+(t3(g,bta,q)/(dtht*dtht)))
return
end

function b(g,bta,dq,q)
real*8 g,bta,b,q,dq,t1,t2
b = (t1(g,bta,q)/(dq*dq))-(t2(g,bta,q)/(2.d0*dq))
return
end

function e(g,bta,dtht,q)
real*8 g,bta,e,q,dtht,t3
e = t3(g,bta,q)/(dtht*dtht)
return
end

function t1(g,bta,q)
real*8 g,bta,q,t1
t1 = (q*q)*(1.d0-(q*q*bta*(g-1.d0)))
return
end

function t2(g,bta,q)
real*8 g,bta,q,t2
t2 = q*(1.d0-(q*q*bta*(g-3.d0)))
```

```
      return
      end

      function t3(g,bta,q)
      real*8 g,bta,q,t3
      t3 = 1.d0-((g+1.d0)*bta*q*q)
      return
      end

*****
***  Thomas Algorithm ... tridiagnol matrix solver
*****

      subroutine sy(il,iu,bb,dd,aa,cc)
      real*8 r
      double precision aa(iu),bb(iu),cc(iu),dd(iu)
      integer il,iu,lp,i,j,nmax
      parameter(nmax=10000)
      double precision f(nmax)

***  making sure that the original matrix
***  stays in tact ... work vector f(nmax)

      do 5  i=il,iu
         f(i) = dd(i)
 5    continue
```

```
*** establish upper triangular matrix

      lp = il+1
      do 10  i=lp,iu
      r = bb(i)/dd(i-1)
      dd(i)=dd(i)-r*aa(i-1)
 10   cc(i) = cc(i)-r*cc(i-1)

***  back substitution

      cc(iu)=cc(iu)/dd(iu)
      do 20  i=lp,iu
      j = iu-i+il
 20   cc(j)=( cc(j)-(aa(j)*cc(j+1)) )/dd(j)

***   solution stored in cc

***   restoring the diagnol of original matrix
      do 25  i=il,iu
         dd(i) = f(i)
 25   continue
      return
      end

*****
```

```
***   Calculates the cosine transform of a set y(1:n+1) of real-valued
***   data points.  The transformed data replace the original data in
***   array y. N must be a power of 2.  This program, without changes,
***   also calculates the inverse cosine transform, but in this case
***   the output array should be multiplied by 2/n.
*****


      subroutine cosft1(y,n)
      integer j,n
      real*8 sum,y1,y2
      real*8 theta,wi,wpi,wpr,wr,wtemp
      double precision y(n+1)
      theta = dacos(-1.d0)/n
      wr = 1.d0
      wi = 0.d0
      wpr = -2.d0*sin(0.5d0*theta)**2
      wpi = dsin(theta)
      sum = 0.5d0*(y(1)-y(n+1))
      y(1) = 0.5d0*(y(1)+y(n+1))
      do 11  j=1,(n/2)-1
         wtemp = wr
         wr = wr*wpr-wi*wpi+wr
         wi = wi*wpr+wtemp*wpi+wi
         y1 = 0.5d0*(y(j+1)+y(n-j+1))
         y2 = (y(j+1)-y(n-j+1))
         y(j+1) = y1-wi*y2
         y(n-j+1) = y1+wi*y2
         sum = sum+wr*y2
11    continue
      call realft(y,n,1)
      y(n+1) = y(2)
      y(2) = sum
      do 12 j=4,n,2
         sum = sum +y(j)
         y(j) = sum
12    continue
      return
      end


      subroutine realft(data,n,isign)
      integer isign,n
      integer i,i1,i2,i3,i4,n2p3
      double precision data(n)
      real*8 c1,c2,h1i,h1r,h2i,h2r,wis,wrs
      real*8 theta,wi,wpi,wpr,wr,wtemp
      theta = dacos(-1.d0)/(dble(n/2))
      c1 = 0.5d0
      if(isign.eq.1) then
         c2 = -0.5d0
         call four1(data,n/2,1)
      else
```

```
    c2 = 0.5d0
    theta = -theta
  endif
  wpr = -2.d0*dsin(0.5d0*theta)**2
  wpi = dsin(theta)
  wr = 1.d0+wpr
  wi = wpi
  n2p3 = n+3
  do 11  i=2,n/4
     i1 = 2*i-1
     i2 = i1+1
     i3 = n2p3-i2
     i4 = i3+1
     wrs = wr
     wis = wi
     h1r = c1*(data(i1)+data(i3))
     h1i = c1*(data(i2)-data(i4))
     h2r = -c2*(data(i2)+data(i4))
     h2i = c2*(data(i1)-data(i3))
     data(i1) = h1r+wrs*h2r-wis*h2i
     data(i2) = h1i+wrs*h2i+wis*h2r
     data(i3) = h1r-wrs*h2r+wis*h2i
     data(i4) = -h1i+wrs*h2i+wis*h2r
     wtemp=wr
     wr = wr*wpr-wi*wpi+wr
     wi = wi*wpr+wtemp*wpi+wi
```

```
11    continue
      if(isign.eq.1) then
         h1r = data(1)
         data(1) = h1r + data(2)
         data(2) = h1r - data(2)
      else
         h1r = data(1)
         data(1)=c1*(h1r+data(2))
         data(2)=c1*(h1r-data(2))
         call four1(data,n/2,-1)
      endif
      return
      end



      subroutine four1(data,nn,isign)
      integer nn,isign
      integer i,istep,j,m,mmax,n
      real*8 tempi,tempr,theta,wi,wpi,wpr,wr,wtemp
      double precision data(2*nn)
      n = 2*nn
      j=1
      do 11  i=1,n,2
         if(j.gt.i) then
            tempr = data(j)
            tempi = data(j+1)
            data(j) = data(i)
            data(j+1) = data(i+1)
            data(i) = tempr
            data(i+1) = tempi
         endif
      m = n/2
1     if((m.ge.2).and.(j.gt.m)) then
      j=j-m
      m=m/2
      goto 1
      endif
      j=j+m
11    continue
      mmax = 2
2     if(n.gt.mmax) then
         istep = 2*mmax
         theta = (2.d0*dacos(-1.d0))/(isign*mmax)
         wpr =  -2.d0*dsin(0.5d0*theta)**2
         wpi = dsin(theta)
         wr = 1.d0
         wi = 0.d0
         do 13  m=1,mmax,2
            do 12  i=m,n,istep
               j=i+mmax
               tempr = wr*data(j)-wi*data(j+1)
               tempi = wr*data(j+1)+wi*data(j)
```

```
              data(j) = data(i)-tempr
              data(j+1) = data(i+1)-tempi
              data(i) = data(i) + tempr
              data(i+1) = data(i+1)+tempi
 12         continue
            wtemp = wr
            wr = wr*wpr-wi*wpi+wr
            wi = wi*wpr+wtemp*wpi+wi
 13       continue
          mmax = istep
          goto 2
          endif
          return
          end

*****
***   This routine employs the LU decomposition routine from the
***   Numerical Recipes handbook and then multiplies the result
***   by the identity matrix to find the inverse of a general
***   NxN matrix.  The inverse of the matrix is stored in Y and
***   the original matrix A is destroyed...
*****

      subroutine matinv(a,n,np,indx,y)
      real d
      integer i,j,n,np
      double precision a(n,n),y(n,n),indx(n)

      do 22  i=1,n
         do 33  j=1,n
            y(i,j)=0.d0
 33        continue
      y(i,i)=1.d0
 22    continue

      call ludcmp(a,n,np,indx,d)

      do 10  j=1,n
       call lubksb(a,n,np,indx,y(1,j))
 10   continue
 56   format(1x,6(f8.4,3x))
      return
      end

***
*****   LU Decomposition
***

      subroutine ludcmp(a,n,np,indx,d)
      real*8 d,aamax,sum,tiny,dum
      integer i,j,n,np,k,nmax,imax
      parameter (nmax=10000,tiny=1.0e-16)
      double precision  a(n,n),indx(n),vv(nmax)
      d = 1.0
      do 12  i=1,n
      aamax = 0.0
      do 11  j=1,n
      if(abs(a(i,j)).gt.aamax) aamax=abs(a(i,j))
 11   continue
      if(aamax.eq.0) pause 'singular matrix'
      vv(i)=1./aamax
 12   continue
      do 19  j=1,n
         do 14  i=1,j-1
      sum = a(i,j)
      do 13  k=1,i-1
         sum = sum - a(i,k)*a(k,j)
 13      continue
      a(i,j) = sum
 14   continue
      aamax = 0.0
      do 16 i=j,n
       sum = a(i,j)
       do 15  k=1,j-1
          sum = sum-a(i,k)*a(k,j)
 15       continue
          a(i,j) = sum
          dum = vv(i)*abs(sum)
```

```
              if(dum.ge.aamax) then
                 imax = i
                 aamax = dum
              endif
 16           continue
           if (j.ne.imax) then
               do 17 k=1,n
                  dum = a(imax,k)
                  a(imax,k)=a(j,k)
                  a(j,k) = dum
 17               continue
           d=-d
           vv(imax) = vv(j)
           endif
           indx(j)=imax
           if(a(j,j).eq.0.) a(j,j) = tiny
           if(j.ne.n) then
              dum = 1./a(j,j)
              do 18  i=j+1,n
                 a(i,j) = a(i,j)*dum
18               continue
           endif
19         continue
           return
           end
```

```
      subroutine lubksb(a,n,np,indx,b)
      real*8 sum
      integer i,j,n,np,ii,ll
      double precision  a(n,n),indx(n),b(n)
      ii = 0
      do 12  i=1,n
         ll = indx(i)
         sum = b(ll)
         b(ll)=b(i)
         if (ii.ne.0) then
       do 11  j=ii,i-1
          sum = sum - a(i,j)*b(j)
11        continue
      else if (sum.ne.0) then
        ii = i
        endif
        b(i) = sum
12      continue
      do 14  i=n,1,-1
         sum = b(i)
         do 13  j=i+1,n
            sum = sum-a(i,j)*b(j)
13          continue
      b(i) = sum/a(i,i)
14    continue
      return
      end

      function psi0(ainf,q,tht)
      real*8 ainf,q,tht,psi0,q2,ainf2
      real*8 arg1,arg2,arg3,arg4
      ainf2 = ainf*ainf
      q2 = q*q
      arg1 = q2+ainf2+2.d0*ainf*q*cos(tht)
      arg2 = q2+ainf2-2.d0*ainf*q*cos(tht)
      arg3 = q2*ainf2+1.d0+2.d0*ainf*q*cos(tht)
      arg4 = q2*ainf2+1.d0-2.d0*ainf*q*cos(tht)
      psi0 = log((arg1*arg2)/(arg3*arg4))
      return
      end

      function bndq0(ainf,minf,tht)
      real*8 minf,ainf,tht,bndq0,pi
      real*8 arg1,arg2,arg3,minf2
      real*8 arg4,arg5,extra
      minf2 = minf*minf
      pi = dacos(-1.d0)
      arg1 = (tht-pi)*(tht-pi)
      arg2 = 1.d0-minf2
      arg3 = arg1/arg2

      arg4 = 32.d0*arg1*arg2
      arg5 = ainf*(-4.d0*arg2-pi*pi)*(-4.d0*arg2-pi*pi)
      extra = arg4/arg5

      bndq0 = (2.d0/(ainf*(1.d0+arg3)))+extra
*      bndq0 = (2.d0/(ainf*(1.d0+arg3)))
      return
      end

      function bndq1(ainf,minf,tht)
      real*8 minf,ainf,tht,bndq1,pi
      real*8 arg1,arg2,arg3
      real*8 ainf2,minf2
      real*8 arg4,extra
      pi = dacos(-1.d0)
      ainf2 = ainf*ainf
      minf2 = minf*minf
      arg1 = (1.d0-ainf)*(1.d0-ainf)
      arg2 = ainf2*(tht-pi)*(tht-pi)
      arg3 = 1.d0-minf2

      arg4 = arg3*arg1 + (ainf2*pi*pi/4.d0)
      extra = arg2/arg4
      bndq1 = -1.d0*log(arg1+(arg2/arg3)) + extra
*      bndq1 = -1.d0*log(arg1+(arg2/arg3))
      return
      end
```

```
function bndtop(ainf,minf,q)
real*8 ainf,minf,q,bndtop,ainf2
real*8 arg1,arg2,arg3,arg4,pi,minf2
pi = dacos(-1.d0)
ainf2 = ainf*ainf
minf2 = minf*minf
arg1 = pi*ainf2
arg2 = (1.d0-minf2)
arg3 = (q-ainf)*(q-ainf)
arg4 = pi*pi*ainf2/4.d0
bndtop = arg1/(arg2*arg3+arg4)
return
end


function force(g,ainf,minf,q,theta)
implicit double precision (a-h,o-z)
real*8 minf,minf2,pi
minf2 = minf*minf
ainf2 = ainf*ainf
pi = dacos(-1.d0)
tht = theta
t2 = ainf**2
t3 = t2**2
t4 = t3*q
```

```
 t5 = theta**2
 t7 = minf**2
 t8 = t7**2
 t9 = t8*t7
 t10 = t3*t9
 t12 = t3*ainf
 t14 = q**2
 t15 = t14**2
 t16 = t15*q
 t19 = pi**2
 t21 = t12*t7
 t23 = t9*t15
 t25 = t12*t8
 t26 = t8*t15
 t29 = t2*ainf
 t30 = t29*t14
 t34 = t29*t9
 t36 = t14*q
 t40 = 4*t4*t5-t10*q+2*t12*t5+2*t9*t16-4*t8*t16+2*t12*t19+3*t21
#+2*t7*t16-5*t23*ainf-2*t12-t25+10*t26*ainf-t21*t5-2*t30-t21
#*t19-t21*g+t25*g+t34*t14+2*t2*t36*t7-8*t4*t7
 t42 = t2*t8
 t44 = t3*t8
 t45 = q*t5
 t47 = t8*t14
 t48 = g*t29
```

```
  t49 = t48*t5
  t51 = t48*t19
  t53 = g*t14
  t55 = t29*t8
  t57 = t7*t15
  t58 = g*ainf
  t61 = t3*t7
  t63 = t2*t9
  t64 = g*t36
  t66 = q*t19
  t68 = t7*t14
  t69 = t29*t19
  t72 = t29*t5
  t78 = 4*t4*t19-5*t42*t36+t44*t45+t47*t49+t47*t51-t34*t53+t55
#*t53+t57*t58+t23*t58-4*t61*t45-t63*t64+t44*t66+t68*t69+4*t4
#+6*t30*t7-t47*t72-t47*t69-t68*t51-5*t55*t14+5*t44*q
  t84 = theta*pi
  t87 = g*q
  t89 = t87*t5
  t91 = t87*t19
  t99 = q*theta*pi
  t111 = -4*t12*theta*pi+3*t63*t36-5*t57*ainf-8*t4*t84-t68*t49
#+t10*t87-t44*t89-t44*t91+t21*g*t5+t21*g*t19+t68*t72+8*t61*t99
#-4*t61*t66+2*t21*t84-2*t2*t7*t64+3*t42*t64+2*t61*t87-3*t44
#*t87+2*t61*t89-4*t61*g*t99
  t120 = t7*t36
```

```
 t121 = g*t2
 t122 = t121*t5
 t126 = t2*theta*pi
 t128 = t121*t19
 t130 = t8*t36
 t138 = t29*theta*pi
 t142 = t2*t5
 t145 = t2*t19
 t152 = 2*t130*t128+2*t68*g*t138-2*t47*g*t138+4*t120*t142
#-8*t120*t126+4*t120*t145-4*t130*t142+8*t130*t126-4*t130
#*t145-2*t68*t138+2*t47*t138
 top=2.d0*(-q+ainf)*(t40+t78+t111+2*t61*t91+2*t44*g*t99
#-2*t21*g*theta*pi-2*t44*t99-2*t26*t58-2*t120*t122+4*t120
#*g*t126-2*t120*t128+2*t130*t122-4*t130*g*t126+t152)

 b1 = ainf2*(2.d0+(g-1.d0)*minf2)
 b2 = 1.d0-minf2
 b3 = (q-ainf)*(q-ainf)
 b4 = (theta-pi)*(theta-pi)
 b5 = (b2*b3+ainf2*b4)
 bot = b1*b5*b5

 if(bot.eq.0.d0) PAUSE 'force is singular'

 f1 = top/bot
 f2 = exforce(g,ainf,minf,q,tht)
```

```
      force = f1+f2
*      force = f1
      return
      end


      function exforce(g,ainf,minf,q,tht)
      implicit double precision (a-h,o-z)
      real*8 minf
      pi = dacos(-1.d0)
      t1 = minf**2
      t2 = t1**2
      t3 = t2*t1
      t4 = q**2
      t5 = t4**2
      t6 = t5*t4
      t7 = t3*t6
      t9 = t1*t6
      t11 = ainf**2
      t12 = t11**2
      t13 = t12*t11
      t14 = t13*t2
      t16 = t11*ainf
      t17 = t4*q
      t18 = t16*t17
      t19 = t18*t1
```

```
t20 = t2*t6
t22 = tht**2
t24 = t12*ainf
t25 = t24*q
t26 = t25*t1
t27 = t12*t3
t29 = pi**2
t30 = t29**2
t32 = t4*t12
t34 = t5*t11
t36 = 128*t7*g+128*t9*g+256*t14*g-3072*t19-256*t20*g-512*t14
#+768*t18*t22-2560*t26+896*t27*t4-64*t25*t30+128*t32*t30-512
#*t34*t29
t37 = t13*t1
t40 = t3*t5
t41 = t22*t11
t43 = tht*pi
t44 = t43*t1
t48 = t2*t5
t49 = g*t22
t50 = t49*t11
t52 = t1*t5
t53 = t52*g
t54 = t43*t11
t56 = t48*g
t58 = g*t29
```

```
 t59 = t58*t11
 t62 = t5*q
 t63 = t3*t62
 t65 = 8*t37*t30-128*t32*t29+1024*t40*t41-4096*t34*t44+1024
#*t40*t29*t11+512*t48*t50+512*t53*t54-1024*t56*t54-192*t52
#*t59-1536*t32-64*t14*t29-512*t63*ainf
 t69 = t22*t1
 t71 = t29*pi
 t72 = tht*t71
 t73 = t72*t11
 t75 = t58*ainf
 t77 = g*t30
 t78 = t77*t11
 t80 = t43*ainf
 t86 = 1024*t25-1536*t18*t43+1024*t34*t43-2304*t18*t69+
#256*t48*t73+640*t37-384*t63*t75+64*t48*t78+768*t63*t80+
#768*t63*g*t80-128*t56*t73+256*t7*t58
 t88 = t49*ainf
 t90 = t41*t29
 t93 = t3*tht*pi
 t95 = t2*tht
 t96 = t95*t71
 t98 = g*tht
 t99 = t98*pi
 t101 = t24*t2
 t102 = t101*q
 t103 = t13*t3
 t106 = t12*t2
 t107 = t106*t4
 t108 = t2*t62
 t110 = 256*t7*t49-384*t63*t88+64*t56*t90+1536*t18*t93-
#128*t32*t96-512*t7*t99+2048*t102-256*t20-128*t103*g+
#128*t7+768*t18*t29-3328*t107+1024*t108*ainf
 t113 = t13*t29
 t116 = t24*t3
 t117 = t116*q
 t119 = t16*t3*t17
 t120 = t22*t29
 t123 = t1*t17
 t124 = t123*g
 t125 = t72*t16
 t127 = t2*t29
 t130 = t2*t17
 t132 = t29*t1
 t134 = 192*t113*t1-256*t25*t22+128*t9-512*t117-1024*t119
#-64*t25*t120+512*t25*t43-64*t124*t125+2432*t18*t127-2624
#*t34*t127-192*t130*t125+256*t25*t132
 t135 = t11*t2
 t137 = t69*t29
 t140 = t130*g
 t142 = t22*t16*t29
 t146 = t77*t16
 t150 = t30*t16
 t154 = -2048*t135*t5-192*t32*t137+192*t123*t125-32*t140
#*t142+64*t140*t125+32*t124*t142+32*t123*t146-2048*t40*t54
#-96*t123*t142-96*t123*t150+96*t130*t142+448*t48*t59
 t159 = t2*t22
 t167 = t159*t29
 t169 = t1*g
 t170 = t169*t22
 t172 = t98*t71
 t174 = -256*t40*t50+512*t40*g*t54-2560*t34*t159+2304*t18
#*t159-256*t40*t59+96*t130*t150-32*t130*t146+640*t25*t69+
#128*t25*t72-32*t25*t167-128*t25*t170+64*t26*t172
 t175 = t2*g
 t176 = t175*t29
 t180 = g*t5
 t183 = g*t4
 t185 = t1*t62
 t186 = g*ainf
 t190 = t16*t2*t17
 t192 = t22*ainf
 t195 = t3*t29
 t198 = 128*t25*t176-64*t102*t172+640*t11*t1*t180-640*t12
#*t1*t183-512*t185*t186+4608*t18*t44+128*t103+3072*t190-256*
#t52*t50-384*t63*t192-128*t48*t90-768*t18*t195+64*t32*t167
 t202 = t30*t11
 t204 = t3*t22
 t207 = t29*ainf
 t214 = g*q
 t216 = t11*t3
 t219 = 128*t52*t202-768*t18*t204+256*t32*t132+768*t108*t207
#-384*t185*t207+1024*t108*t186-512*t63*t186+128*t52*t90-640
#*t27*t183+512*t116*t214+896*t216*t5-1024*t101*t214
 t226 = t32*t1
 t232 = t3*g
 t233 = t232*t29
 t235 = -1280*t135*t180+512*t24*t1*t214+640*t216*t180+1280
#*t106*t183+768*t108*t192+3968*t226-512*t185*ainf-512*t34*t22
#+64*t25*t96-64*t37*t58+64*t14*t58-128*t25*t233
 t237 = t175*t30
 t241 = t232*t22
 t243 = t49*t29
 t245 = t169*t30
 t247 = t95*pi
 t250 = t1*tht*t71
 t252 = t175*t22
 t257 = 32*t25*t237-256*t25*t93+256*t117*t99-128*t25*t241
#+32*t102*t243-32*t25*t245+1024*t25*t247-192*t25*t250+256
#*t25*t252-32*t26*t243+256*t26*t99-512*t102*t99
 t260 = t1*t30
 t264 = t2*t30
 t273 = -384*t185*t192-8*t37*t77+96*t25*t260+128*t25*t204
#+128*t25*t195-32*t25*t264-1280*t25*t44+96*t25*t137-512*t25
```

```
#*t159-384*t25*t127-184*t32*t260-4608*t18*t247+5120*t34*t247
 t289 = 72*t32*t245-128*t32*t127+384*t18*t169*t29-896*t18
#*t176-128*t226*t172+2048*t190*t99-1024*t19*t99-256*t32*t72
#-64*t32*t237+1408*t34*t1+64*t226*t243-1024*t18*t252
 t305 = 512*t18*t170-128*t48*t202+128*t32*t120-384*t63*t207
#-64*t52*t78+768*t185*t80-1536*t108*t80+768*t108*t75-512*t20
#*t58-384*t185*t75+128*t53*t73-1536*t108*g*t80+768*t185*g*t80
 t319 = 1024*t20*t99-64*t53*t90-512*t9*t99+64*t32*t264+768
#*t108*t88-384*t185*t88-512*t20*t49+384*t32*t250+256*t9*t49
#-128*t37*g+2048*t34*t69+2112*t34*t132
 t329 = -64*t107*t243-1024*t119*t99-256*t52*t73+512*t18*t241
#-2432*t18*t132-16*t13*t30-256*t34+1024*t18-256*t13-128*t113
#+128*t107*t172+512*t18*t233+256*t9*t58
 t333 = t36+t65+t86+t110+t134+t154+t174+t198+t219+t235+t257
#+t273+t289+t305+t319+t329
 top = t333


 t1 = minf**2
 t4 = q**2
 t6 = ainf**2
 t11 = pi**2
 t13 = -4*t4+8*q*ainf-4*t6+4*t1*t4-8*t1*q*ainf+4*t1*t6-t6*t11
 t14 = t13**2
 t16 = (2+t1*g-t1)*t14*t13

bot = t16
```

```
if(bot.eq.0.d0) PAUSE 'extraforce is singular'

exforce = top/bot
return
end


function pert(ainf,minf,q,theta)
real*8 pert,ainf,minf,q,theta,psi1
real*8 t1,t2,t3,t5,t6,t8,t10,t11,t12,t13
real*8 t15,t16,t18,t19,t20,t21,t23,t24,t25
real*8 t26,t27,t29,t40,t43,top,bottom,tol,psi0
tol = 1e-16
t1 = q**2
t2 = ainf**2
t3 = t1*t2
t5 = dcos(theta)
t6 = ainf*q*t5
t10 = t1**2
t11 = t10**2
t12 = t11*t2
t13 = t5**2
t15 = t2**2
t16 = t1*t15
t18 = t10*t1
```

```
 t19 = t18*t15
 t20 = t10*t2
 t21 = t15*t2
 t23 = t15**2
 t24 = t18*t23
 t25 = t11*t21
 t26 = t23*t1
 t27 = t18*t21
 t29 = -2*t12*t13+t16+t11*t15-t19-2*t3+t20+t12-t10*t21
#-2*t24-2*t25+t26+2*t27+t23*t10
 t40 = -4*t27*t13+4*t3*t13-2*t26*t13+2*t25*t13+2*t19*t13+2*
#t24*t13+2*t21*t13*t10-2*t16*t13-2*t20*t13+t18-t15+t21-t10
 t43 = -(t3+1+2*t6)*(t3+1-2*t6)*(t29+t40)
 top = t43

 t1 = ainf**2
 t2 = q**2
 t3 = t1*t2
 t6 = ainf*q*dcos(theta)
 t8 = (t3+1+2*t6)**2
 t11 = (t3+1-2*t6)**2
 t16 = t1*t8*t11*(t2+t1+2*t6)*(t2+t1-2*t6)
 bottom = t16

 if(bottom.le.tol) then
    write(*,*) 'q= ',q
```

```
   write(*,*) 'theta= ',theta
   PAUSE 'denomenator is zero'
else
endif

psi1 = (top/bottom)
pert = psi0(ainf,q,theta)+(minf*minf*psi1)
return
end

subroutine closure(n,m,ainf,minf,g,dq,dtht,psi)
real*8 ainf,minf,g,q,tht,dq,dtht,qfix,ff,q0,qn
real*8 pi,d1,d2,d3,d4,d5,d6,d7,eps,qtest
real*8 sum,f0,fn,total1,diff,total2
real*8 h0,hn,minf2,ainf2,rho,t1,dqqtpsi
real*8 tol,dpdq,dpdt,dpdq0,dpdt0,dpdqn,dpdtn,hh,tht0,thtn
real*8 coff0,coffn,coff,t10,t1n,thtfix,tf,w1,w2
real*8 dm,dm1,dm2,d8,ddm,gtol
parameter (eps=0.1d0,tol=1e-16)
integer j1,j2
integer i,j,n,m,is1,ieps,i0,in,j0,jn,jeps
double precision psi(n,m)
pi = dacos(-1.d0)
ainf2 = ainf * ainf
minf2 = minf*minf
is1 = (ainf/dq)+1
```

```
      ieps = ((ainf+eps)/dq)+1
      d6 = 0.d0
      gtol = (1.d0/(n-1))*(1.d0/(n-1))

***
***  integration along path 1 ... tht=pi & q=ainf+eps..1
***
      j=m
      q0 = ainf+eps
      qn = 1.d0
      i0 = ieps
      in = n
      qtest = (i0-1)*dq
      diff = abs(qtest-q0)
      if(diff.gt.tol) then
      write(*,*) 'q0= ',q0
      write(*,*) 'qtest= ',qtest
      PAUSE 'starting point error on path 1'
      else
      endif
      f0 = -sin(pi)*((-3.d0*psi(i0,j)+4.d0*psi(i0+1,j)-psi(i0+2,j))
     */(2.d0*dq))/(rho(g,ainf,minf,q0)*q0)
      fn = -sin(pi)*((3.d0*psi(in,j)-4.d0*psi(in-1,j)+psi(in-2,j))
     */(2.d0*dq))/(rho(g,ainf,minf,qn)*qn)
      sum = 0.d0
      do 5  i=i0+1,n-1
```

```
         q = (i-1)*dq
      ff = sin(pi)*((psi(i+1,j)-psi(i-1,j))/(2.d0*dq))
     */(rho(g,ainf,minf,q)*q)
         sum = ff + sum
 5    continue
      d1 = -dq*(f0+fn+2.d0*sum)/2.d0
      write(*,*) 'result of path1=  ',d1


***
***  integration along path 2 ... q=ainf+eps & tht=pi-eps..pi
***
      qfix = ainf+eps
      i = i0
      j0 = m-(i0-is1)
      jn = m
      jeps = j0
      tht0 = (pi/2.d0)+(j0-1)*dtht
      thtn = pi
      dpdq0 = ((psi(i0+1,j0)-psi(i0-1,j0))/(2.d0*dq))
      dpdt0=(-3.d0*psi(i0,j0)+4.d0*psi(i0,j0+1)-psi(i0,j0+2))
     */(2.d0*dtht)
      dpdqn=((psi(i0+1,jn)-psi(i0-1,jn))/(2.d0*dq))
      dpdtn=(3.d0*psi(i0,jn)-4.d0*psi(i0,jn-1)+psi(i0,jn-2))
     */(2.d0*dtht)
      h0 = (cos(tht0)*dpdq0/(rho(g,ainf,minf,qfix))) -
     * (sin(tht0)*dpdt0/(rho(g,ainf,minf,qfix)*qfix))
```

```
      hn = (cos(thtn)*dpdqn/(rho(g,ainf,minf,qfix))) -
     * (sin(thtn)*dpdtn/(rho(g,ainf,minf,qfix)*qfix))
      sum = 0.d0
      do 6  j = j0+1,m-1
      tht = (pi/2.d0)+(j-1)*dtht
      dpdq = ((psi(i0+1,j)-psi(i0-1,j))/(2.d0*dq))
      dpdt = ((psi(i0,j+1)-psi(i0,j-1))/(2.d0*dtht))
      hh = (cos(tht)*dpdq/rho(g,ainf,minf,qfix)) -
     * (sin(tht)*dpdt/(rho(g,ainf,minf,qfix)*qfix))
      sum = sum + hh
 6    continue
      d2 = -dtht*(h0+hn+2.d0*sum)/2.d0
      write(*,*) 'result of path2= ',d2

***
***  integration along path 3 ... tht=pi-eps & q=ainf-eps..ainf+eps
***
      j=j0
      thtfix = (pi/2.d0)+(j-1)*dtht
      q0 = ainf-eps
      qn = ainf+eps
      i0 = is1-(ieps-is1)
      in = ieps
      qtest = (i0-1)*dq
      diff = abs(qtest-q0)
      if(diff.gt.tol) then
 write(*,*) 'q0= ',q0
 write(*,*) 'qtest= ',qtest
 PAUSE 'starting point error on path 3'
 else
 endif
 dpdt0 = ((psi(i0,j+1)-psi(i0,j-1))/(2.d0*dtht))
 dpdq0=((psi(i0+1,j)-psi(i0-1,j))/(2.d0*dq))
 dpdtn = ((psi(in,j+1)-psi(in,j-1))/(2.d0*dtht))
 dpdqn=((psi(in+1,j)-psi(in-1,j))/(2.d0*dq))
 coff0 = 1.d0+((g-1.d0)/2.d0)*minf2*(1.d0-(q0*q0/ainf2))
 t10 = (minf2/ainf2)/(rho(g,ainf,minf,q0)*coff0)
 coffn = 1.d0+((g-1.d0)/2.d0)*minf2*(1.d0-(qn*qn/ainf2))
 t1n = (minf2/ainf2)/(rho(g,ainf,minf,qn)*coffn)
 tf = thtfix
 w1 = cos(tf)*dpdt0*t10-(cos(tf)*dpdt0/(rho(g,ainf,minf,q0)
#*q0*q0))
 w2 = sin(tf)*dpdq0/(rho(g,ainf,minf,q0)*q0)
 f0 = w1-w2
 w1 = cos(tf)*dpdtn*t1n-(cos(tf)*dpdtn/(rho(g,ainf,minf,qn)
#*qn*qn))
 w2 = sin(tf)*dpdqn/(rho(g,ainf,minf,qn)*qn)
 fn = w1-w2
 sum = 0.d0
 do 7  i = i0+1,in-1
    q = (i-1)*dq
 coff = 1.d0+((g-1.d0)/2.d0)*minf2*(1.d0-(q*q/ainf2))
```

```
      t1 = (minf2/ainf2)/(rho(g,ainf,minf,q)*coff)
      dpdq = (psi(i+1,j)-psi(i-1,j))/(2.d0*dq)
      dpdt = ((psi(i,j+1)-psi(i,j-1))/(2.d0*dtht))
      ff = (t1*cos(thtfix)*dpdt)-(cos(thtfix)*dpdt/
     #(rho(g,ainf,minf,q)*q*q))-(sin(thtfix)*dpdq/
     #(rho(g,ainf,minf,q)*q))
      sum = sum + ff
 7    continue
      d3 = -dq*(f0+fn+2.d0*sum)/2.d0
      write(*,*) 'result of path3 = ',d3

***
***  integration along path 4 ...q=ainf-eps & tht=Pi-eps..Pi
***

      qfix = ainf-eps
      i = i0
      j0 = jeps
      jn = m
      tht0 = (pi/2.d0)+(j0-1)*dtht
      thtn = pi
      dpdq0=((psi(i0+1,j0)-psi(i0-1,j0))/(2.d0*dq))
      dpdt0=(-3.d0*psi(i0,j0)+4.d0*psi(i0,j0+1)-psi(i0,j0+2))
     */(2.d0*dtht)
      dpdqn=((psi(i0+1,jn)-psi(i0-1,jn))/(2.d0*dq))
      dpdtn=(3.d0*psi(i0,jn)-4.d0*psi(i0,jn-1)+psi(i0,jn-2))
     */(2.d0*dtht)
      h0 = (cos(tht0)*dpdq0/(rho(g,ainf,minf,qfix))) -
     * (sin(tht0)*dpdt0/(rho(g,ainf,minf,qfix)*qfix))
      hn = (cos(thtn)*dpdqn/(rho(g,ainf,minf,qfix))) -
     * (sin(thtn)*dpdtn/(rho(g,ainf,minf,qfix)*qfix))
      sum = 0.d0
      do 8  j=j0+1,jn-1
      tht = (pi/2.d0)+(j-1)*dtht
      dpdq = ((psi(i0+1,j)-psi(i0-1,j))/(2.d0*dq))
      dpdt = ((psi(i0,j+1)-psi(i0,j-1))/(2.d0*dtht))
      hh = (cos(tht)*dpdq/rho(g,ainf,minf,qfix)) -
     * (sin(tht)*dpdt/(rho(g,ainf,minf,qfix)*qfix))
      sum = sum + hh
 8    continue
      d4 = dtht*(h0+hn+2.d0*sum)/2.d0
      write(*,*) 'result of path4 = ',d4

***
***   integration along path 5 ...tht=pi and q=0..ainf-eps
***
      j=m
      q0 = 0.d0
      qn = ainf-eps
      in = i0
      dm=(2.d0*psi(1,j)-5.d0*psi(2,j)+4.d0*psi(3,j)-psi(4,j))/(dq*dq)
      j1 = j-1
      dm1=(2.d0*psi(1,j1)-5.d0*psi(2,j1)+4.d0*psi(3,j1)-psi(4,j1))
      dm1 = dm1/(dq*dq)
      j2 = j1-1
      dm2=(2.d0*psi(1,j2)-5.d0*psi(2,j2)+4.d0*psi(3,j2)-psi(4,j2))
      dm2 = dm2/(dq*dq)
      dqqtpsi=(3.d0*dm-4.d0*dm1+dm2)/(2.d0*dtht)
      dqqtpsi = dqqtpsi/10.d0
*     if(abs(dqqtpsi).gt.gtol) pause 'something wrong at q=0 path 5'
      f0 = -sin(pi)*dm/(rho(g,ainf,minf,q0))
      fn = -sin(pi)*((3.d0*psi(in,j)-4.d0*psi(in-1,j)+psi(in-2,j))
     */(2.d0*dq))/(rho(g,ainf,minf,qn)*qn)
      sum = 0.d0
      do 9  i=2,in-1
         q = (i-1)*dq
      ff = sin(pi)*((psi(i+1,j)-psi(i-1,j))/(2.d0*dq))
     */(rho(g,ainf,minf,q)*q)
         sum = ff + sum
 9    continue
      d5 = -dq*(f0+fn+2.d0*sum)/2.d0
      write(*,*) 'result of path5= ',d5

***
*  evaluating the total contribution at infinity
***
      total1 = d1+d2+d3+d4+d5
      write(*,*)
      write(*,*) 'eps= ',eps
      write(*,*) 'the contribution at infinity= ',total1
      write(*,*)

***
****** integration along path 7
***
      q0 = 0.d0
      qn = 1.d0
      thtfix = pi/2.d0
      tf = thtfix
      ddm=(35.d0*psi(1,1)-104.d0*psi(2,1)+114.d0*psi(3,1)-
     #56.d0*psi(4,1)+11.d0*psi(5,1))/(12.d0*dq*dq)
      write(*,*) 'ddm= ',ddm
      dm=(2.d0*psi(1,1)-5.d0*psi(2,1)+4.d0*psi(3,1)-psi(4,1))/(dq*dq)
      dm1=(2.d0*psi(1,2)-5.d0*psi(2,2)+4.d0*psi(3,2)-psi(4,2))/(dq*dq)
      dm2=(2.d0*psi(1,3)-5.d0*psi(2,3)+4.d0*psi(3,3)-psi(4,3))/(dq*dq)
      dqqtpsi=(-3.d0*dm+4.d0*dm1-dm2)/(2.d0*dtht)
      dqqtpsi = dqqtpsi/10.d0
*     if(abs(dqqtpsi).gt.gtol) pause 'something wrong at q=0 path 7'
      f0=-sin(tf)*ddm/(rho(g,ainf,minf,q0))
      fn=-sin(tf)*((3.d0*psi(n,1)-4.d0*psi(n-1,1)+psi(n-2,1))/(2.d0*dq))
     #/rho(g,ainf,minf,qn)
      sum=0.d0
      do 10  i=2,n-1
         q = (i-1)*dq
```

```
           dpdq = ((psi(i+1,1)-psi(i-1,1))/(2.d0*dq))
           ff = -sin(tf)*dpdq/(rho(g,ainf,minf,q)*q)
           sum = sum + ff
 10     continue
        d7 = dq*(f0+fn+2.d0*sum)/2.d0
        write(*,*) 'result of path7= ',d7


***
*****  integration along path 8 ... q=1 & tht=pi/2..pi
***


      qfix = 1.d0
      j0 = 1
      jn = m
      tht0 = pi/2.d0
      thtn = pi
      dpdq0=(3.d0*psi(n,j0)-4.d0*psi(n-1,j0)+psi(n-2,j0))/(2.d0*dq)
      dpdt0=(-3.d0*psi(n,j0)+4.d0*psi(n,j0+1)-psi(n,j0+2))
     */(2.d0*dtht)
      dpdqn=(3.d0*psi(n,jn)-4.d0*psi(n-1,jn)+psi(n-2,jn))/(2.d0*dq)
      dpdtn=(3.d0*psi(n,jn)-4.d0*psi(n,jn-1)+psi(n,jn-2))/(2.d0*dtht)
      h0 = (cos(tht0)*dpdq0/(rho(g,ainf,minf,qfix))) -
     * (sin(tht0)*dpdt0/(rho(g,ainf,minf,qfix)*qfix))
      hn = (cos(thtn)*dpdqn/(rho(g,ainf,minf,qfix))) -
     * (sin(thtn)*dpdtn/(rho(g,ainf,minf,qfix)*qfix))
      sum = 0.d0
      do 11  j=2,m-1
      tht = (pi/2.d0)+(j-1)*dtht
      dpdq = ((3.d0*psi(n,j)-4.d0*psi(n-1,j)+psi(n-2,j))/(2.d0*dq))
      dpdt = ((psi(n,j+1)-psi(n,j-1))/(2.d0*dtht))
      hh = (cos(tht)*dpdq/rho(g,ainf,minf,qfix)) -
     * (sin(tht)*dpdt/(rho(g,ainf,minf,qfix)*qfix))
      sum = sum + hh
11    continue
      d8 = dtht*(h0+hn+2.d0*sum)/2.d0
      write(*,*) 'result of path8= ',d8
      write(*,*)
      total2 = d7+d8
      write(*,*) 'total at bottom= ',total2
      write(*,*)
      diff = (total2+total1)/total2
      write(*,*) 'around the plane= ',diff
      end

      function rho(g,ainf,minf,q)
      real*8 rho,minf,ainf,q,g,term,coff
      coff = (g-1.d0)/2.d0
      term = 1.d0 + (coff*minf*minf)*(1.d0-(q*q/(ainf*ainf)))
      rho = term**(1.d0/(g-1.d0))
      return
      end
      subroutine cusp(psi,n,m,dq,dtht,ainf,minf,g)
      real*8 dq,dtht,ainf,minf,q,ainf2,minf2,g
      real*8 arg1,arg2,arg3,arg4,term1,dtpsi
      real*8 dqpsi,dtpsi2,dqpsi2,tht,pi,dqpsin,dtpsin
      integer i,j,n,m
      double precision psi(n,m)
      pi = dacos(-1.d0)
      minf2 = minf*minf
      ainf2 = ainf*ainf
      arg1 = (g-1.d0)*minf2/2.d0
      arg2 = minf2/ainf2
*****
***  checking the interior
*****
      do 2  i=2,n-1
      do 5  j=2,m-1
         q = (i-1)*dq
         tht = (pi/2.d0)+(j-1)*dtht
      dtpsi= (psi(i,j+1)-psi(i,j-1))/(2.d0*dtht)
      dqpsi=(psi(i+1,j)-psi(i-1,j))/(2.d0*dq)
      dtpsi2 = dtpsi*dtpsi
      dqpsi2 = dqpsi*dqpsi
      term1 = (1.d0-(q*q/(ainf2)))
      arg3 = (arg2*q*q/(1.d0+arg1*term1))-1.d0
      arg4 = arg3*dtpsi2-q*q*dqpsi2
      if(arg4.gt.0.d0) then
```

```
            write(*,*) 'q= ',q
            write(*,*) 'tht= ',tht
          PAUSE 'cuspidal behaviour '
          else
        endif
 5      continue
 2      continue

*****
***  looking at the equation need not worry about q=0
***  hence, just calculating the value of delta at q=1
*****

      do 10  j=2,m-1
      tht = (pi/2.d0)+(j-1)*dtht
      dqpsin=(3.d0*psi(n,j)-4.d0*psi(n-1,j)+psi(n-2,j))/(2.d0*dq)
      dtpsin=(psi(n,j+1)-psi(n,j-1))/(2.d0*dtht)
      arg3 = ((arg2/(1.d0+arg1*(1-(1/ainf2))))-1.d0)
      arg4 = arg3*dtpsin*dtpsin-dqpsin*dqpsin
      if(arg4.gt.0.d0) then
         write(*,*) 'q= ',1.d0
         write(*,*) 'tht= ',tht
       PAUSE 'cuspidal behaviour '
       else
      endif
10    continue
```

```
*****
***  Looking at the equation the solution should be O.K.
***  on the boundaries since dtpsi=0
*****
      return
      end



      subroutine map(g,ainf,minf,psi,n,m,x,y,nn,k)
      real*8 q,tht,dq,dtht,fx,pi,cx,sumx
      real*8 g,ainf,minf,ainf2,minf2,dqn,qn
      real*8 cn,c0,fy,sumy,cy,yint,xint,ys
      real*8 dvx0,dvxn,dvx,dvy0,dvyn,dvy,psi
      real*8 bx1,bx2,by1,by2,dqpsi,dtpsi,rho
      integer i,j,n,m,i0,in,i1,k,nn,j1,j2,ijunk
      integer istop1,istop2,itotal,jtotal
      parameter(ys=15.d0)
      double precision x(nn,m),y(nn,m),psi(n,m)
      open(69,file='grid.out')
      write(*,*) 'nn= ',nn
      pi = dacos(-1.d0)
      ainf2 = ainf*ainf
      minf2 = minf*minf
      dq = 1.d0/(n-1)
      dtht = (pi/2.d0)/(m-1)
```

```
*****
***  the stagnation point needs to be treated specially
***  t1 = 2*dtpsidt/q*q = d3psidq2dt
***  t2 = dpsidq/q = d2psidq2
*****

      do 2  j=1,m
         do 1  i1=1,nn
      x(i1,j) = 0.d0
      y(i1,j) = 0.d0
 1    continue
 2    continue

      do 3  j=1,m
       i0 = 1
      do 5  i1=2,nn
       in = i0 + k

***
*  integration of the x-direction
***

      sumx = 0.d0
      c0 = fx(psi,n,m,i0,j,ainf,minf,g)
      cn = fx(psi,n,m,in,j,ainf,minf,g)
      do 10  i=i0+1,in-1
      q = (i-1)*dq
      tht = (pi/2.d0)+(j-1)*dtht
      cx = fx(psi,n,m,i,j,ainf,minf,g)
      sumx = sumx + cx
 10   continue
      xint = (c0+cn+2.d0*sumx)*(dq/2.d0)
      x(i1,j) = xint + x(i1-1,j)

***
*  integration of the y-direction
***

      sumy = 0.d0
      c0 = fy(psi,n,m,i0,j,ainf,minf,g)
      cn = fy(psi,n,m,in,j,ainf,minf,g)
      do 12  i=i0+1,in-1
      q = (i-1)*dq
      tht = (pi/2.d0)+(j-1)*dtht
      cy = fy(psi,n,m,i,j,ainf,minf,g)
      sumy = sumy + cy
 12   continue
      yint = (c0+cn+2.d0*sumy)*(dq/2.d0)
      y(i1,j) = yint + y(i1-1,j)
      i0 = i0 + k
      dqn = 1.d0/(nn-1)
      qn = (i1-1)*dqn
      if(y(i1,m).gt.ys) go to 50
 5    continue
 3    continue
 300  format(1x,i3,2x,i3,2x,2(f12.6,2x))

***
*  stepping on to the vortex boundary @ q=1
***
 50   istop1 = i1
      i = n
      j1 = m-1
      q = (i-1)*dq
      tht = (pi/2.d0)+(j1-1)*dtht
      bx1 = cos(tht)*dqpsi(psi,n,m,i,j1)/rho(g,ainf,minf,q)
      bx2 = sin(tht)*dtpsi(psi,n,m,i,j1)/(rho(g,ainf,minf,q)*q)
      dvx0= bx1-bx2

      by1 = sin(tht)*dqpsi(psi,n,m,i,j1)/rho(g,ainf,minf,q)
      by2 = cos(tht)*dtpsi(psi,n,m,i,j1)/(rho(g,ainf,minf,q)*q)
      dvy0 = by1+by2

      j2 = m
      q = (i-1)*dq
      tht = (pi/2.d0)+(j2-1)*dtht
      bx1 = cos(tht)*dqpsi(psi,n,m,i,j2)/rho(g,ainf,minf,q)
```

```
bx2 = sin(tht)*dtpsi(psi,n,m,i,j2)/(rho(g,ainf,minf,q)*q)
dvxn = bx1-bx2


by1 = sin(tht)*dqpsi(psi,n,m,i,j2)/rho(g,ainf,minf,q)
by2 = cos(tht)*dtpsi(psi,n,m,i,j2)/(rho(g,ainf,minf,q)*q)
dvyn = by1+by2


dvx = dq*(dvx0+dvxn)/2.d0
dvy = dq*(dvxn+dvyn)/2.d0
write(*,*) 'dvx= ',dvx
write(*,*) 'dvy= ',dvy
write(*,*) nn,m-1,'  x(nn,m-1)= ',x(nn,m-1)
write(*,*) nn,m-1,'  y(nn,m-1)= ',y(nn,m-1)
x(nn,m) = x(nn,m-1)+dvx
y(nn,m) = y(nn,m-1)+dvy
write(*,*) 'xb= ',x(nn,m)
write(*,*) 'yb= ',y(nn,m)
j = m
i0 = n
do 42  i1=nn-1,1,-1
   in = i0-k
   sumx = 0.d0
   c0 = fx(psi,n,m,i0,j,ainf,minf,g)
   cn = fx(psi,n,m,in,j,ainf,minf,g)
do 34  i=i0-1,in+1,-1
  q = (i-1)*dq
```

```
      tht = (pi/2.d0)+(j-1)*dtht
      cx = fx(psi,n,m,i,j,ainf,minf,g)
      sumx = sumx + cx
 34   continue
      xint = (c0+cn+2.d0*sumx)*(dq/2.d0)
      x(i1,j) = x(i1+1,j)-xint


      sumy = 0.d0
      c0 = fy(psi,n,m,i0,j,ainf,minf,g)
      cn = fy(psi,n,m,in,j,ainf,minf,g)
      do 33  i=i0-1,in+1,-1
        q = (i-1)*dq
        tht = (pi/2.d0)+(j-1)*dtht
        cy = fy(psi,n,m,i,j,ainf,minf,g)
        sumy = sumy + cy
 33   continue
      yint = (c0+cn+2.d0*sumy)*(dq/2.d0)
      y(i1,j) = y(i1+1,j)-yint


      i0 = i0 - k
      if(abs(y(i1,j)).gt.ys) go to 40
 42   continue
 40   q=(i-1)*dq
      istop2 = i1
      write(*,*) 'istop2= ',istop2,'  q=',q
      write(*,*) 'istop1= ',istop1
```

```
***
*  formatting the output
***


      ijunk = (istop2-istop1+1)


      itotal = nn-ijunk
      jtotal = m
      write(*,*) 'itotal= ',itotal
      write(*,*) 'jtotal= ',jtotal


***
*  the header required by Tecplot
***


      write(69,*) 'TITLE = "Incompressible Flow"'
      write(69,*) 'VARIABLES = x, y, psi'
      write(69,*) 'ZONE T="ZONE-one", I=',itotal,' , J=',m,
     #', F=POINT'


      do 22  j=1,m
      do 21  i1=1,nn
         if(i1.ge.istop1.and.i1.le.istop2) go to 21
      dqn = 1.d0/(nn-1)
      q = (i1-1)*dqn
      tht = (pi/2.d0)+(j-1)*dtht
      i = (i1-1)*k + 1
      write(69,301) x(i1,j),y(i1,j),psi(i,j)
21    continue
22    continue
301   format(1x,3(f8.4,3x))
      return
      end


      function fx(psi,n,m,i,j,ainf,minf,g)
      real*8 fx,q,tht,dq,dtht,ainf,minf,g,fx0
      real*8 pi,arg1,arg2,den,rho,dtpsi,dqpsi
      integer i,j,n,m
      double precision psi(n,m)
      pi = dacos(-1.d0)
      dq = 1.d0/(n-1)
      dtht = (pi/2.d0)/(m-1)
      q = (i-1)*dq
      if(q.eq.0) then
         fx = fx0(psi,n,m,i,j,ainf,minf,g)
         go to 12
      else
       endif
      tht = (pi/2.d0)+(j-1)*dtht
      arg1 = cos(tht)*dtpsi(psi,n,m,i,j)*den(g,ainf,minf,q)
      arg2 = sin(tht)*dqpsi(psi,n,m,i,j)/(rho(g,ainf,minf,q)*q)
      fx = arg1-arg2
```

```
12   return
     end

     function fy(psi,n,m,i,j,ainf,minf,g)
     real*8 fy,q,tht,dq,dtht,ainf,minf,g,fy0
     real*8 pi,arg1,arg2,den,rho,dtpsi,dqpsi
     integer i,j,n,m
     double precision psi(n,m)
     pi = dacos(-1.d0)
     dq = 1.d0/(n-1)
     dtht = (pi/2.d0)/(m-1)
     q = (i-1)*dq
     if(q.eq.0) then
        fy = fy0(psi,n,m,i,j,ainf,minf,g)
        go to 13
        else
        endif
     tht = (pi/2.d0)+(j-1)*dtht
     arg1 = sin(tht)*dtpsi(psi,n,m,i,j)*den(g,ainf,minf,q)
     arg2 = cos(tht)*dqpsi(psi,n,m,i,j)/(rho(g,ainf,minf,q)*q)
     fy = arg1+arg2
13   return
     end


     function fx0(psi,n,m,i,j,ainf,minf,g)
```

```
real*8 ainf,minf,g,fx0,ainf2,minf2,dq,dtht
real*8 arg1,arg2,arg3,cst,tt1,tt2,t3,pi
real*8 rho,dtpsi,q,tht
integer i,j,n,m
double precision psi(n,m)
pi = dacos(-1.d0)
ainf2 = ainf*ainf
minf2 = minf*minf
dq = 1.d0/(n-1)
dtht = (pi/2.d0)/(m-1)
q = (i-1)*dq
tht = (pi/2.d0)+(j-1)*dtht
t3 = 1.d0+((g-1.d0)/2.d0)*minf2*(1.d0-(q*q/(ainf2)))
cst = minf2/(ainf2*rho(g,ainf,minf,q)*t3)
arg1 = cos(tht)*dtpsi(psi,n,m,i,j)*cst
arg2 = cos(tht)*tt1(psi,n,m,j)/rho(g,ainf,minf,q)
arg3 = sin(tht)*tt2(psi,n,m,j)/(rho(g,ainf,minf,q))
fx0  = arg1-arg2-arg3
return
end

function fy0(psi,n,m,i,j,ainf,minf,g)
real*8 ainf,minf,g,fy0,ainf2,minf2,dq
real*8 arg1,arg2,arg3,tt1,tt2,t3,cst,dtht
real*8 rho,dtpsi,q,tht,pi
integer i,j,n,m
```

```
double precision psi(n,m)
pi = dacos(-1.d0)
dq = 1.d0/(n-1)
dtht = (pi/2.d0)/(m-1)
ainf2 = ainf*ainf
minf2 = minf*minf
q=(i-1)*dq
tht = (pi/2.d0)+(j-1)*dtht
t3 = 1.d0+((g-1.d0)/2.d0)*minf2*(1.d0-(q*q/(ainf2)))
cst = minf2/(ainf2*rho(g,ainf,minf,q)*t3)
arg1 = sin(tht)*dtpsi(psi,n,m,i,j)*cst
arg2 = sin(tht)*tt1(psi,n,m,j)/rho(g,ainf,minf,q)
arg3 = cos(tht)*tt2(psi,n,m,j)/(rho(g,ainf,minf,q))
fy0 = arg1-arg2+arg3
return
end


function tt2(psi,n,m,j)
real*8 dq,dtht,tt2,pi
integer i,j,n,m
double precision psi(n,m)
pi = dacos(-1.d0)
dq = 1.d0/(n-1)
dtht = (pi/2.d0)/(m-1)
i=1
```

```
  tt2 = (2.d0*psi(i,j)-5.d0*psi(i+1,j)+4.d0*psi(i+2,j)
 #-psi(i+3,j))/(dq*dq)
  return
  end


  function tt1(psi,n,m,j)
  real*8 dq,dtht,tt1,tht,pi,tt2
  integer j,n,m
  double precision psi(n,m)
  pi = dacos(-1.d0)
  dq = 1.d0/(n-1)
  dtht = (pi/2.d0)/(m-1)
  tht = (pi/2.d0)+(j-1)*dtht
  if(tht.eq.(pi/2.d0)) then
  tt1 = (-3.d0*tt2(psi,n,m,1)+4.d0*tt2(psi,n,m,2)
 #-tt2(psi,n,m,3))/(2.d0*dtht)
  else
  if(tht.eq.pi) then
  tt1 = (3.d0*tt2(psi,n,m,j)-4.d0*tt2(psi,n,m,j-1)
 #+tt2(psi,n,m,j-2))/(2.d0*dtht)
  else
  tt1 = (tt2(psi,n,m,j+1)-tt2(psi,n,m,j-1))/(2.d0*dtht)
  endif
  endif
  return
  end
```

```
function den(g,ainf,minf,q)
real*8 g,ainf,minf,q,den,q2,rho
real*8 arg1,arg2,arg3,ainf2,minf2
ainf2 = ainf*ainf
minf2 = minf*minf
q2 = q*q
arg1= 1.d0+((g-1.d0)/2.d0)*minf2*(1.d0-(q2/ainf2))
arg2 = minf2/(ainf2*arg1*rho(g,ainf,minf,q))
arg3 = 1.d0/(q*q*rho(g,ainf,minf,q))
den = arg2-arg3
return
end

function dtpsi(psi,n,m,i,j)
real*8 dq,dtht,dtpsi,pi
integer i,j,n,m
double precision psi(n,m)
pi = dacos(-1.d0)
dq = 1.d0/(n-1)
dtht = (pi/2.d0)/(m-1)
if(j.eq.(m-1).or.j.eq.m) then
dtpsi = (3.d0*psi(i,j)-4.d0*psi(i,j-1)+psi(i,j-2))/(2.d0*dtht)
else
if(j.eq.1) then
dtpsi = (-3.d0*psi(i,j)+4.d0*psi(i,j+1)-psi(i,j+2))/(2.d0*dtht)
```

```
else
dtpsi = (psi(i,j+1)-psi(i,j-1))/(2.d0*dtht)
endif
endif
return
end

function dqpsi(psi,n,m,i,j)
real*8 dq,dtht,dqpsi,q,pi
integer i,j,n,m
double precision psi(n,m)
pi = dacos(-1.d0)
dq = 1.d0/(n-1)
dtht = (pi/2.d0)/(m-1)
q = (i-1)*dq
if(q.eq.1) then
dqpsi = (3.d0*psi(i,j)-4.d0*psi(i-1,j)+psi(i-2,j))/(2.d0*dq)
else
dqpsi = (psi(i+1,j)-psi(i-1,j))/(2.d0*dq)
endif
return
end
```

Appendix E

Fortran listing for Stuart-type vortices

```
*****
***   This program solves for the compressible analogue of the
***   Stuart Vortices.  The flow is assumed to be Homentropic
***   and the governing equations linearized for iterative solver.
***   Newton's method is employed for the unknown psi(i,j)
***   h = wall height
*****

      real*8 x,y,mach,mach2,g,h,psiwall,eps,eps2,dx2,dy2,tol
      real*8 tip1j,tij,tim1j,tijp1,tijm1,tip1jp1,tip1jm1,rhoc
      real*8 tim1jp1,tim1jm1,cp2dx,cp1dx,cpmd,cp2dy,cp1dy,crhs,cp
      real*8 chk1,chk2,eqnorm,deltrho,rcond,dmach,sol,maxerr,h2
      real*8 dervx,psinc,chk,dervy,value,circ,dh,omega
      real*8 xmax,ymax,umax,vmax,b1,b2
      integer i,j,ieqn,kiter,kmax,kplc,kp,ichk,itol
      integer j1,j2,j3,j4,j5,j6,j7,j8,j9,ijob,kmach,kmachmax
      integer lda,ml,mu,job,iflag,iflag2,iflagh,khmax,kh
      include 'input.h'
      parameter(mu=2*n,ml=2*n,lda=2*ml+mu+1,job=0,kmax=400,khmax=0)
      parameter(kmachmax=0)
      common /param/ g,mach,eps,h,psiwall
      double precision psi(n,m),rho(n,m),dpsi(n,m),drho(n,m)
      double precision abd(lda,nm),z(nm),rhsvec(nm),fint(n)
      double precision u(n,m),v(n,m)

      integer ipvt(nm)


      open(69,file='psi.contour')
      open(70,file='rho.contour')
      open(71,file='init.guess')
      open(72,file='reg.soln')
      open(73,file='search.out')
      open(74,file='vel.profile')
      open(75,file='omega.contour')
      open(76,file='info.out')
      open(77,file='vel.geom')

      do 811  i=1,lda
```

```
          do 82  j=1,nm
             abd(i,j) = 0.d0
             rhsvec(j) = 0.d0
 82       continue
 811   continue

*
*** initializing the parameters of the problem
*** for incomp. flow:  psinf = log(kappa/2)
*

        mach = 0.d0
        h = 2.d0
        h2 = h*h
        dh = 0.d0
        g = 1.4d0
        eps = 0.8d0
        eps2 = eps*eps
        psiwall = log(cosh(h))
        tol = 1e-8
        dx2 = dx*dx
        dy2 = dy*dy
        mach2 = mach*mach

        write(*,*)
        write(*,*) 'n= ',n
```

```
      write(*,*) 'm= ',m
      write(*,*) 'mach= ',mach
      write(*,*) 'eps= ',eps
      write(*,*) 'wall height= ',h
      write(*,*) 'psi wall= ',psiwall
      write(*,*)
      kiter = 0
      kmach = 0


*
*** Initial Guess ... close to solution
*


      do 5  i=1,n
       x = (i-1)*dx
        do 10  j=1,m
         y = (j-1)*dy
        ieqn = n*(j-1)+i
*        psi(i,j) = psinc(x,y,eps,h)
*        rho(i,j) = 1.d0
       read(71,*) psi(i,j),rho(i,j)
 10   continue
 5    continue

*     do 7  i=1,n
*         psi(i,m) = psiwall
* 7    continue

      if(kiter.eq.0) go to 901

*
***  the iteration procedure begins
*

 911  kiter = kiter + 1

      do 439  i=1,n
         do 440  j=1,m
            ieqn = n*(j-1)+i
            dpsi(i,j) = rhsvec(ieqn)
440      continue
439   continue

      do 441  i=1,n
         do 442  j=1,m
            drho(i,j) = deltrho(psi,rho,dpsi,i,j)
442      continue
441   continue

      do 222  i=1,n
         do 223  j=1,m
```

```
            ieqn = n*(j-1)+i
            psi(i,j) = psi(i,j)+dpsi(i,j)
            rho(i,j) = rho(i,j)+drho(i,j)
 223      continue
 222   continue


***
*****   checking the infinity norm of the equations
***

      ijob = 1
      chk1 = eqnorm(psi,rho,ijob)
      ijob = 2
      chk2 = eqnorm(psi,rho,ijob)
      write(*,*) 'chk1= ',chk1
      write(*,*) 'chk2= ',chk2
      write(*,*)
      write(*,*)
      write(*,*)
*
*** applying the stencil to the grid
*** and initializing the rhs-vector
*
 901  do 20  i=2,n-1
         do 25  j=2,m-1
```

```
        ieqn = n*(j-1)+i
        x = (i-1)*dx
        y = (j-1)*dy

        tip1j = (cp2dx(psi,rho,i,j)/dx2)+
      #(cp1dx(psi,rho,i,j)/(2.d0*dx))
        tij = cp(psi,rho,i,j)-2.d0*( (cp2dx(psi,rho,i,j)/dx2)
      #+ (cp2dy(psi,rho,i,j)/dy2))
        tim1j =(cp2dx(psi,rho,i,j)/dx2)-(cp1dx(psi,rho,i,j)/(2.d0*dx))
        tijp1=(cp2dy(psi,rho,i,j)/dy2)+(cp1dy(psi,rho,i,j)/(2.d0*dy))
        tijm1=(cp2dy(psi,rho,i,j)/dy2)-(cp1dy(psi,rho,i,j)/(2.d0*dy))
        tip1jp1 = cpmd(psi,rho,i,j)/(4.d0*dx*dy)
        tim1jm1 = cpmd(psi,rho,i,j)/(4.d0*dx*dy)
        tip1jm1 = -cpmd(psi,rho,i,j)/(4.d0*dx*dy)
        tim1jp1 = -cpmd(psi,rho,i,j)/(4.d0*dx*dy)

       j1 = n*(j-1)+i
       kp = kplc(ml,mu,nm,ieqn,j1)
       abd(kp,j1) = tij
*       abd(kp,j1) = tij/tij

       j2 = n*(j-1)+i+1
       kp = kplc(ml,mu,nm,ieqn,j2)
       abd(kp,j2) = tip1j
*       abd(kp,j2) = tip1j/tij
```

```
      j3 = n*(j-1)+i-1
      kp = kplc(ml,mu,nm,ieqn,j3)
      abd(kp,j3) = tim1j
*       abd(kp,j3) = tim1j/tij

      j4 = n*(j)+i
      kp = kplc(ml,mu,nm,ieqn,j4)
      abd(kp,j4) = tijp1
*       abd(kp,j4) = tijp1/tij

      j5 = n*(j-2)+i
      kp = kplc(ml,mu,nm,ieqn,j5)
      abd(kp,j5) = tijm1
*       abd(kp,j5) = tijm1/tij

      j6 = n*(j)+i+1
      kp = kplc(ml,mu,nm,ieqn,j6)
      abd(kp,j6) = tip1jp1
*       abd(kp,j6) = tip1jp1/tij

      j7 = n*(j-2)+i-1
      kp = kplc(ml,mu,nm,ieqn,j7)
      abd(kp,j7) = tim1jm1
*       abd(kp,j7) = tim1jm1/tij
```

```
         j8 = n*(j-2)+i+1
         kp = kplc(ml,mu,nm,ieqn,j8)
         abd(kp,j8) = tip1jm1
*         abd(kp,j8) = tip1jm1/tij

         j9 = n*(j)+i-1
         kp = kplc(ml,mu,nm,ieqn,j9)
         abd(kp,j9) = tim1jp1
*          abd(kp,j9) = tim1jp1/tij
*
         rhsvec(ieqn) = -crhs(psi,rho,i,j)
*         rhsvec(ieqn) = -crhs(psi,rho,i,j)/tij
*
 25      continue
 20      continue

*
*** setting the boundary conditions
*** for coefficient matrix AND rhs-vector
*
*** at x=0
*
       i = 1
       do 30  j=1,m
          ieqn = n*(j-1)+i
          x = (i-1)*dx
```

```
        y = (j-1)*dy

      j1 = ieqn
      kp = kplc(ml,mu,nm,ieqn,j1)
      abd(kp,j1) = -3.d0
*      abd(kp,j1) = -3.d0/(-3.d0)

      j2 = ieqn+1
      kp = kplc(ml,mu,nm,ieqn,j2)
      abd(kp,j2) = 4.d0
*      abd(kp,j2) = 4.d0/(-3.d0)

      j3 = ieqn+2
      kp = kplc(ml,mu,nm,ieqn,j3)
      abd(kp,j3) = -1.d0
*     abd(kp,j3) = -1.d0/(-3.d0)
*
      rhsvec(ieqn) = -2.d0*dx*dervx(psi,i,j)
*      rhsvec(ieqn) = -2.d0*dx*dervx(psi,i,j)/(-3.d0)
*
 30   continue
*
*** at x=pi
*
      i=n
      do 31  j=1,m
```

```
              ieqn = n*(j-1)+i
              x = (i-1)*dx
              y = (j-1)*dy
          j1 = ieqn
          kp = kplc(ml,mu,nm,ieqn,j1)
          abd(kp,j1) = 3.d0
*          abd(kp,j1) = 3.d0/3.d0

          j2 = ieqn-1
          kp = kplc(ml,mu,nm,ieqn,j2)
          abd(kp,j2) = -4.d0
*          abd(kp,j2) = -4.d0/3.d0

          j3 = ieqn-2
          kp = kplc(ml,mu,nm,ieqn,j3)
          abd(kp,j3) = 1.d0
*          abd(kp,j3) = 1.d0/3.d0

*
          rhsvec(ieqn) = -2.d0*dx*dervx(psi,i,j)
*          rhsvec(ieqn) = -2.d0*dx*dervx(psi,i,j)/3.d0
*
 31       continue
*
*** at y=0
*
```

```
      j=1
      do 32  i=2,n-1
         ieqn = n*(j-1)+i
         x = (i-1)*dx
         y = (j-1)*dy
      j1 = ieqn
      kp = kplc(ml,mu,nm,ieqn,j1)
      abd(kp,j1) = -3.d0
*      abd(kp,j1) = -3.d0/(-3.d0)



      j2 = n*(j)+i
      kp = kplc(ml,mu,nm,ieqn,j2)
      abd(kp,j2) = 4.d0
*      abd(kp,j2) = 4.d0/(-3.d0)

      j3 = n*(j+1)+i
      kp = kplc(ml,mu,nm,ieqn,j3)
      abd(kp,j3) = -1.d0
*      abd(kp,j3) = -1.d0/(-3.d0)

*
      rhsvec(ieqn) = -2.d0*dy*dervy(psi,i,j)
*      rhsvec(ieqn) = -2.d0*dy*dervy(psi,i,j)/(-3.d0)
*
 32   continue
```

```
*
*** at y=1
*
      j=m
      do 33  i=2,n-1
         ieqn = n*(j-1)+i
         x = (i-1)*dx
         y = (j-1)*dy
      j1 = ieqn
      kp = kplc(ml,mu,nm,ieqn,j1)
      abd(kp,j1) = 1.d0
*
         rhsvec(ieqn) = psiwall-psi(i,j)
*
 33   continue

*
***  Solving the problem A.X=B
*
      call dgbco(abd,lda,nm,ml,mu,ipvt,rcond,z)

      call dgbsl(abd,lda,nm,ml,mu,ipvt,rhsvec,job)

         do 101  i=1,lda
            do 102  j=1,nm
```

```
                  abd(i,j) = 0.d0
  102          continue
  101      continue


*
***  checking the error with L infinity norm
*
        maxerr = 0.d0
        do 50  i=1,nm
           chk = abs(rhsvec(i))
        if(chk.gt.maxerr)  then
          maxerr=chk
          ichk = i
          else
          endif
50     continue
       write(*,*) 'iteration no.= ',kiter
       write(*,*) 'mach= ',mach
       write(*,*) 'condition number= ',rcond
       write(*,*) 'maxerr= ',maxerr
       write(*,*) 'ichk= ',ichk
       if(maxerr.gt.tol) then
          if(kiter.gt.kmax) PAUSE 'kmax exceeded'
          go to 911
       else
       write(*,*) 'procedure done in ',kiter,' steps'
```

```
      endif

      ijob = 1
      chk1 = eqnorm(psi,rho,ijob)
      ijob = 2
      chk2 = eqnorm(psi,rho,ijob)
      write(*,*) 'chk1= ',chk1
      write(*,*) 'chk2= ',chk2
      write(*,*)

      value = circ(psi,rho,fint)/h
      write(*,*)
      write(*,*) 'circulation in the domain= ',value
      write(*,*) 'core density= ', rho(n,1)
      write(*,*)

      call core(psi,rho,xmax,vmax,ymax,umax)
      rhoc = rho(n,1)
      b1 = ymax
      b2 = pi-xmax
      write(76,341) mach,value,b1,b2,rhoc,kiter
      call flush(76)
 341  format(1x,5(f10.5,2x),5i)
      write(77,343) h,mach,umax,vmax
 343  format(1x,4(f11.5,2x))
*      write(*,*) 'do you want to increase the Mach number? '
*      write(*,*) 'press 1 for yes'
*      write(*,*) 'press 2 for no'
*      read(*,*) iflag2
*      write(*,*)
      iflag2 = 1

      write(*,*)
*      write(*,*) 'do you want to change the tolerance?'
*      write(*,*) 'press 1 for yes'
*      write(*,*) 'press 2 for no '
*      read(*,*) itol
      itol = 2

      if(itol.eq.1) then
         write(*,*) 'what is the new tolerance? '
         read(*,*) tol
         else
      endif

      if(iflag2.eq.1) then
*         write(*,*) 'what is the mach number increment?'
*         read(*,*) dmach
         dmach = 0.01d0

      if(kmach.ge.kmachmax) go to 1010
      mach = mach + dmach
      mach2 = mach*mach
      kmach = kmach + 1
      kiter = 0

*       if(mach.ge.0.1d0) go to 1010
      go to 901
      else
      endif

      iflagh = 1
*      iflagh = 0
      if(iflagh.eq.1) then
         write(73,241) mach,h,value,kiter
         call flush(73)
 241     format(1x,3(f12.6,3x),i6)
         write(*,*) mach,h,value
         if(kh.ge.khmax) go to 1010

         h = h + dh
         kh = kh + 1
         kiter = 0
         go to 901
      else
      endif
```

```
*
***  formatting the output
*
 1010 write(*,*) 'the output files are for the following'
      write(*,*) 'mach= ',mach
      write(*,*) 'eps= ',eps
      write(*,*) 'wall height= ',h
      write(*,*) 'n= ',n
      write(*,*) 'm= ',m

      do 123  i=1,n
         do 124  j=1,m
            write(72,*) psi(i,j),rho(i,j)
 124     continue
 123  continue


*      write(*,*) 'press 1 if you want streamlines'
*      write(*,*) 'press 0 if you want to bypass'
*      read(*,*) iflag
      iflag = 1
      if(iflag.eq.0) go to 212
      write(69,*) 'TITLE="Stuart Vortex Problem" '
      write(69,*) 'VARIABLES = x, y, psi'
      write(69,*) 'ZONE T="ZONE-one",', 'I= ',n,', J= ',m,' ,F=POINT'
      write(70,*) 'TITLE="Stuart Vortex Problem" '
      write(70,*) 'VARIABLES = x, y, rho'
      write(70,*) 'ZONE T="ZONE-one",', 'I= ',n,', J= ',m,' ,F=POINT'
      write(75,*) 'TITLE="Stuart Vortex Problem" '
      write(75,*) 'VARIABLES = x, y, omega'
      write(75,*) 'ZONE T="ZONE-one",', 'I= ',n,', J= ',m,' ,F=POINT'

      do 111  j=1,m
         do 112  i=1,n
            x = (i-1)*dx
            y = (j-1)*dy
            sol = psi(i,j)
            omega = rho(i,j)*(1.d0-eps2)*exp(-2.d0*psi(i,j))
            omega = omega/(rho(n,m))
          u(i,j) = dervy(psi,i,j)/(h*rho(i,j))
          v(i,j) = -dervx(psi,i,j)/rho(i,j)
             write(69,*) x,y,sol
           write(70,*) x,y,rho(i,j)
           write(75,*) x,y,omega
112      continue
111   continue

      j=m
      do 2  i=1,n
         x = (i-1)*dx
         write(74,*) x,u(i,j),v(i,j)
2     continue
```

```
 212  stop
      end

      subroutine core(psi,rho,xmax,vmax,ymax,umax)
      real*8 xmax,umax,vmax,ymax,dervx,dervy,u,v,x,y,rhoc
      integer i,j
      include 'input.h'
      double precision psi(n,m),rho(n,m)

      rhoc = rho(n,m)
      vmax = 0.d0
      j=1
      do 20  i=1,n
         x = (i-1)*dx
         v = -rhoc*dervx(psi,i,j)/rho(i,j)
      if(v.gt.vmax) then
         vmax = v
         xmax = x
      else
      endif
 20   continue

      umax = 0.d0
      i=n
      do 30  j=1,m
      y=(j-1)*dy
      u = rhoc*dervy(psi,i,j)/rho(i,j)
   if(u.gt.umax) then
      umax = u
      ymax = y
   else
   endif
30 continue
   return
   end


   function kplc(ml,mu,n2,i,j)
   integer ml,mu,n2,j,kplc,i,ms
   ms = ml+mu+1
   kplc = i-j+ms
   return
   end


   function eqnorm(psi,rho,ijob)
   real*8 eqnorm,fc,t1,max,sc
   integer i,j,ijob
   include 'input.h'
   double precision psi(n,m),rho(n,m)
```

```
      max = 0.d0
      do 10  i=2,n-1
         do 12  j=2,m-1

            if(ijob.eq.1) then
            t1 = abs(fc(psi,rho,i,j))
          else
            t1 = abs(sc(psi,rho,i,j))
            endif

            if(t1.gt.max) then
               max = t1
            else
            endif
12       continue
10    continue

      eqnorm = max
      return
      end


      function circ(psi,rho,fint)
      real*8 circ,dervy,sum,vcirc
      integer i,j
      include 'input.h'
      double precision psi(n,m),rho(n,m)
      double precision fint(n)

      do 5  i=1,n
         fint(i) = 0.d0
 5    continue

      j=m
      do 10  i=1,n
         fint(i) = dervy(psi,i,j)/rho(i,j)
 10   continue

      sum = 0.d0
      do 12  i=2,n-1
         sum = fint(i)+sum
 12   continue

      vcirc = fint(1)+fint(n)+2.d0*sum
      circ = vcirc*dx/2.d0
      return
      end

******
***  matrix coefficients evaluated as functions
***    ... coefficients of the stretched Newton variables
******
```

```
function cp2dx(psi,rho,i,j)
real*8 a,gf,r,q,t1,t2,cp2dx
integer i,j
include 'input.h'
double precision psi(n,m),rho(n,m)

t1 = a(psi,rho,i,j)
t2 = gf(psi,rho,i,j)*r(psi,rho,i,j)/q(psi,rho,i,j)
cp2dx = t1-t2
return
end



function cp1dx(psi,rho,i,j)
real*8 t1,t2,t3,t4,t5,t6,t7,t8,t9,cp1dx
real*8 q,q2,qx,qy,c,f,r,gf,rx,t,hf,ry
integer i,j
include 'input.h'
double precision psi(n,m),rho(n,m)

q2 = q(psi,rho,i,j)*q(psi,rho,i,j)
t1 = c(psi,rho,i,j)
t2 = f(psi,rho,i,j)*r(psi,rho,i,j)/q(psi,rho,i,j)
t3 = gf(psi,rho,i,j)
t4 = r(psi,rho,i,j)*qx(psi,rho,i,j)/q2
```

```
t5 = rx(psi,rho,i,j)/q(psi,rho,i,j)
t6 = t(psi,rho,i,j)/q(psi,rho,i,j)
t7 = hf(psi,rho,i,j)
t8 = r(psi,rho,i,j)*qy(psi,rho,i,j)/q2
t9 = ry(psi,rho,i,j)/q(psi,rho,i,j)
cp1dx = t1-t2+t3*(t4-t5-t6)+t7*(t8-t9)
return
end



function cpmd(psi,rho,i,j)
real*8 t1,t2,cpmd
real*8 s,gf,q,hf,r
integer i,j
include 'input.h'
double precision psi(n,m),rho(n,m)

t1 = -s(psi,rho,i,j)*gf(psi,rho,i,j)/q(psi,rho,i,j)
t2 = -hf(psi,rho,i,j)*r(psi,rho,i,j)/q(psi,rho,i,j)
cpmd = t1+t2
return
end

function cp2dy(psi,rho,i,j)
real*8 t1,t2,cp2dy,b,hf,s,q
integer i,j
```

```
include 'input.h'
double precision psi(n,m),rho(n,m)

t1 = b(psi,rho,i,j)
t2 = hf(psi,rho,i,j)*s(psi,rho,i,j)/q(psi,rho,i,j)
cp2dy = t1-t2
return
end

function cp1dy(psi,rho,i,j)
real*8 t1,t2,t3,t4,t5,t6,t7,t8,t9,cp1dy
real*8 q,q2,qx,qy,d,f,gf,s,sx,sy,t,hf
integer i,j
include 'input.h'
double precision psi(n,m),rho(n,m)

q2 = q(psi,rho,i,j)*q(psi,rho,i,j)
t1 = d(psi,rho,i,j)
t2 = f(psi,rho,i,j)*s(psi,rho,i,j)/q(psi,rho,i,j)
t3 = gf(psi,rho,i,j)
t4 = qx(psi,rho,i,j)*s(psi,rho,i,j)/q2
t5 = sx(psi,rho,i,j)/q(psi,rho,i,j)
t6 = hf(psi,rho,i,j)
t7 = s(psi,rho,i,j)*qy(psi,rho,i,j)/q2
t8 = sy(psi,rho,i,j)/q(psi,rho,i,j)
t9 = t(psi,rho,i,j)/q(psi,rho,i,j)
```

```
cp1dy = t1-t2+t3*(t4-t5)+t6*(t7-t8-t9)
return
end



function cp(psi,rho,i,j)
real*8 t1,t2,t3,t4,t5,t6,t7,t8,cp
real*8 ty,tx,q,q2,e,f,gf
real*8 t,qx,hf,qy
integer i,j
include 'input.h'
double precision psi(n,m),rho(n,m)

q2 = q(psi,rho,i,j)*q(psi,rho,i,j)
t1 = e(psi,rho,i,j)
t2 = f(psi,rho,i,j)*t(psi,rho,i,j)/q(psi,rho,i,j)
t3 = gf(psi,rho,i,j)
t4 = qx(psi,rho,i,j)*t(psi,rho,i,j)/q2
t5 = tx(psi,rho,i,j)/q(psi,rho,i,j)
t6 = hf(psi,rho,i,j)
t7 = t(psi,rho,i,j)*qy(psi,rho,i,j)/q2
t8 = ty(psi,rho,i,j)/q(psi,rho,i,j)
cp = t1-t2+t3*(t4-t5)+t6*(t7-t8)
return
end
```

```
      function crhs(psi,rho,i,j)
      real*8 t1,t2,t3,t4,t5,t6,t7,t8,crhs
      real*8 fc,f,sc,q,gf,qx,q2,scx,scy,qy,hf
      integer i,j
      include 'input.h'
      double precision psi(n,m),rho(n,m)

      q2 = q(psi,rho,i,j)*q(psi,rho,i,j)
      t1 = fc(psi,rho,i,j)
      t2 = f(psi,rho,i,j)*sc(psi,rho,i,j)/q(psi,rho,i,j)
      t3 = gf(psi,rho,i,j)
      t4 = sc(psi,rho,i,j)*qx(psi,rho,i,j)/q2
      t5 = scx(psi,rho,i,j)/q(psi,rho,i,j)
      t6 = hf(psi,rho,i,j)
      t7 = sc(psi,rho,i,j)*qy(psi,rho,i,j)/q2
      t8 = scy(psi,rho,i,j)/q(psi,rho,i,j)
      crhs = t1-t2+(t3*(t4-t5))+(t6*(t7-t8))
      return
      end

******
***  the functions that give the matrix coefficients
******

      function fc(psi,rho,i,j)
      real*8 fc,dervx,dervy,rho2,h,h2,eps
```

```
real*8 t1,t2,t3,t4,t5,t6,t7,eps2,t41
real*8 g,mach,psiwall
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

h2 = h*h
eps2 = eps*eps
rho2 = rho(i,j)*rho(i,j)
t1 = -(dervx(rho,i,j)/(rho2))*dervx(psi,i,j)
t2 = (psi(i+1,j)-2.d0*psi(i,j)+psi(i-1,j))/(dx*dx)
t3 = (t2/rho(i,j))+t1
t4 = (psi(i,j+1)-2.d0*psi(i,j)+psi(i,j-1))/(dy*dy)
t41 = t4/(rho(i,j)*h2)
t5 = -dervy(rho,i,j)*dervy(psi,i,j)/(rho2*h2)
t6 = t41+t5
t7 = -rho(i,j)*(1.d0-eps2)*exp(-2.d0*psi(i,j))
fc = t3+t6+t7
return
end


function sc(psi,rho,i,j)
real*8 g,mach,mach2,h,h2,eps,eps2
real*8 sc,dervx,dervy,psiwall
```

```
real*8 t1,t2,t3,t4,t5,t6,t7,t8,t9
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

mach2 = mach*mach
h2 = h*h
eps2 = eps*eps
t1 = rho(i,j)**(g-1.d0)
t2 = 1.d0+((g-1.d0)*mach2/2.d0)
t3 = (g-1.d0)*mach2/(2.d0*rho(i,j)*rho(i,j))
t4 = dervx(psi,i,j)*dervx(psi,i,j)
t5 = dervy(psi,i,j)*dervy(psi,i,j)/h2
t6 = t3*(t4+t5)
t7 = (g-1.d0)*mach2*(1.d0-eps2)/2.d0
t8 = exp(-2.d0*psi(i,j))-exp(-2*psiwall)
t9 = t7*t8
sc = t1-t2+t6+t9
return
end

function a(psi,rho,i,j)
real*8 a
integer i,j
include 'input.h'
```

```
double precision psi(n,m),rho(n,m)

a = 1.d0/rho(i,j)
return
end

function b(psi,rho,i,j)
real*8 b,rho2,h,h2,g,mach,eps,psiwall
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

h2 = h*h
rho2 = rho(i,j)*rho(i,j)
b = 1.d0/(h2*rho2)
return
end

function c(psi,rho,i,j)
real*8 c,dervx,rho2
integer i,j
include 'input.h'
double precision psi(n,m),rho(n,m)

rho2 = rho(i,j)*rho(i,j)
```

```
c = -dervx(rho,i,j)/rho2
return
end

function d(psi,rho,i,j)
real*8 d,dervy,h,h2,rho2,g,mach,eps,psiwall
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

h2 = h*h
rho2 = rho(i,j)*rho(i,j)
d = -dervy(rho,i,j)/(h2*rho2)
return
end

function e(psi,rho,i,j)
real*8 e,eps,eps2,g,mach,h,psiwall
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

eps2 = eps*eps
e = 2.d0*rho(i,j)*(1.d0-eps2)*exp(-2.d0*psi(i,j))
```

```
return
end

function f(psi,rho,i,j)
real*8 f,t1,t2,t3,t4,t5,t6,t7,rho3,g,mach
real*8 dervx,dervy,rho2,h,h2,eps,eps2,psiwall
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

rho2 = rho(i,j)*rho(i,j)
rho3 = rho(i,j)*rho2
h2 = h*h
eps2 = eps*eps
t1 = (psi(i+1,j)-2.d0*psi(i,j)+psi(i-1,j))/(dx*dx)
t2 = -t1/rho2
t3 = 2.d0*dervx(rho,i,j)*dervx(psi,i,j)/rho3
t4 = (psi(i,j+1)-2.d0*psi(i,j)+psi(i,j-1))/(dy*dy)
t5 = t4/(rho2*h2)
t6 = 2.d0*dervy(rho,i,j)*dervy(psi,i,j)/(h2*rho3)
t7 = (1.d0-eps2)*exp(-2.d0*psi(i,j))
f = t2+t3-t5+t6-t7
return
end
```

```
function gf(psi,rho,i,j)
real*8 gf,dervx,rho2
integer i,j
include 'input.h'
double precision psi(n,m),rho(n,m)

rho2 = rho(i,j)*rho(i,j)
gf = -dervx(psi,i,j)/rho2
return
end


function hf(psi,rho,i,j)
real*8 hf,h,h2,dervy,rho2,mach,eps,psiwall,g
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

rho2 = rho(i,j)*rho(i,j)
h2 = h*h
hf = -dervy(psi,i,j)/(rho2*h2)
return
end
```

```
function q(psi,rho,i,j)
real*8 q,h,h2,dervx,dervy,rho3,eps,psiwall
real*8 g,mach,mach2,t1,t2,t3,t4,t5
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

mach2 = mach*mach
h2 = h*h
rho3 = rho(i,j)*rho(i,j)*rho(i,j)
t1 = (g-1.d0)*(rho(i,j)**(g-2.d0))
t2 = (g-1.d0)*mach2/rho3
t3 = dervx(psi,i,j)*dervx(psi,i,j)
t4 = dervy(psi,i,j)*dervy(psi,i,j)/h2
t5 = t2*(t3+t4)
q = t1-t5
return
end

function r(psi,rho,i,j)
real*8 r,dervx,rho2,eps,h
real*8 g,mach,mach2,psiwall
integer i,j
include 'input.h'
```

```
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

rho2 = rho(i,j)*rho(i,j)
mach2 = mach*mach
r = (g-1.d0)*mach2*dervx(psi,i,j)/rho2
return
end

function s(psi,rho,i,j)
real*8 s,h,h2,dervy,rho2,eps
real*8 mach,mach2,g,psiwall
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

mach2 = mach*mach
h2 = h*h
rho2 = rho(i,j)*rho(i,j)
s = (g-1.d0)*mach2*dervy(psi,i,j)/(h2*rho2)
return
end

function t(psi,rho,i,j)
real*8 t,eps,eps2,mach,mach2,g,psiwall,h
```

```
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

mach2 = mach*mach
eps2 = eps*eps
t = -(g-1.d0)*mach2*(1.d0-eps2)*exp(-2*psi(i,j))
return
end



function qx(psi,rho,i,j)
real*8 qx,g,mach,mach2,h,h2,rho3,dervy
real*8 dervx,rho4,t1,t2,t3,t4,t5,t6
real*8 t7,t8,t9,t10,psiwall,eps
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

mach2 = mach*mach
h2 = h*h
rho3 = rho(i,j)*rho(i,j)*rho(i,j)
rho4 = rho(i,j)*rho(i,j)*rho(i,j)*rho(i,j)
t1 = (rho(i,j)**(g-3.d0))
```

```
t2 = (g-1.d0)*(g-2.d0)*dervx(rho,i,j)*t1
t3 = 3.d0*dervx(rho,i,j)*(g-1.d0)*mach2/rho4
t4 = dervx(psi,i,j)*dervx(psi,i,j)
t5 = dervy(psi,i,j)*dervy(psi,i,j)/h2
t6 = t3*(t4+t5)
t7 = 2.d0*(g-1.d0)*mach2/rho3
t8 = (dervx(psi,i,j+1)-dervx(psi,i,j-1))/(2.d0*dy)
t9 = (psi(i+1,j)-2.d0*psi(i,j)+psi(i-1,j))/(dx*dx)
t10 = t7*(dervx(psi,i,j)*t9+dervy(psi,i,j)*t8/h2)
qx = t2+t6-t10
return
end


function qy(psi,rho,i,j)
real*8 qy,mach,mach2,h,h2,rho3,rho4,dervx,psiwall
real*8 t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,dervy,g,eps
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

mach2 = mach*mach
h2 = h*h
rho3 = rho(i,j)*rho(i,j)*rho(i,j)
rho4 = rho(i,j)*rho3
```

```
t1 = rho(i,j)**(g-3.d0)
t2 = (g-1.d0)*(g-2.d0)*dervy(rho,i,j)*t1
t3 = 3.d0*dervy(rho,i,j)*(g-1.d0)*mach2/rho4
t4 = dervx(psi,i,j)*dervx(psi,i,j)
t5 = dervy(psi,i,j)*dervy(psi,i,j)/h2
t6 = t3*(t4+t5)
t7 = 2.d0*(g-1.d0)*mach2/rho3
t8 = (dervx(psi,i,j+1)-dervx(psi,i,j-1))/(2.d0*dy)
t9 = (psi(i,j+1)-2.d0*psi(i,j)+psi(i,j-1))/(dy*dy)
t10 = t7*(t8*dervx(psi,i,j)+t9*dervy(psi,i,j)/h2)
qy = t2+t6-t10
return
end


function rx(psi,rho,i,j)
real*8 rx,mach,mach2,h,h2,dervx,rho2,rho3
real*8 t1,t2,t3,psiwall,eps,g
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

mach2 = mach*mach
h2 = h*h
rho2 = rho(i,j)*rho(i,j)
```

```
rho3 = rho2*rho(i,j)
t1 = -2.d0*dervx(rho,i,j)*(g-1.d0)*mach2*dervx(psi,i,j)/rho3
t2 = (psi(i+1,j)-2.d0*psi(i,j)+psi(i-1,j))/(dx*dx)
t3 = (g-1.d0)*mach2*t2/rho2
rx = t1+t3
return
end



function ry(psi,rho,i,j)
real*8 ry,mach,mach2,h,h2,dervx,dervy,eps
real*8 rho2,rho3,t1,t2,t3,t4,t5,psiwall,g
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

mach2 = mach*mach
h2 = h*h
rho2 = rho(i,j)*rho(i,j)
rho3 = rho2*rho(i,j)
t1 = -2.d0*(g-1.d0)*mach2*dervy(rho,i,j)
t2 = t1*dervx(psi,i,j)/rho3
t3 = (g-1)*mach2/rho2
t4 = (dervx(psi,i,j+1)-dervx(psi,i,j-1))/(2.d0*dy)
t5 = t3*t4
```

```
ry = t2+t5
return
end



function sx(psi,rho,i,j)
real*8 sx,mach,mach2,h,h2,rho2,rho3,dervx
real*8 t1,t2,t3,t4,t5,psiwall,g,eps,dervy
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

mach2 = mach*mach
h2 = h*h
rho2 = rho(i,j)*rho(i,j)
rho3 = rho(i,j)*rho2
t1 = -2.d0*dervx(rho,i,j)*(g-1.d0)*mach2/h2
t2 = t1*dervy(psi,i,j)/rho3
t3 = (g-1.d0)*mach2/(h2*rho2)
t4 = (dervx(psi,i,j+1)-dervx(psi,i,j-1))/(2.d0*dy)
t5 = t3*t4
sx = t2+t5
return
end
```

```
function sy(psi,rho,i,j)
real*8 sy,mach,mach2,h,h2,rho3,rho2,g
real*8 t1,t2,t3,t4,t5,dervy,psiwall,eps
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

mach2 = mach*mach
h2 = h*h
rho2 = rho(i,j)*rho(i,j)
rho3 = rho(i,j)*rho2
t1 = -2.d0*dervy(rho,i,j)*(g-1.d0)*mach2/rho3
t2 = t1*dervy(psi,i,j)/h2
t3 = (g-1.d0)*mach2/(h2*rho2)
t4 = (psi(i,j+1)-2.d0*psi(i,j)+psi(i,j-1))/(dy*dy)
t5 = t3*t4
sy = t5+t2
return
end


function tx(psi,rho,i,j)
real*8 tx,mach,mach2,h,h2,eps,eps2,g
real*8 t1,dervx,psiwall
```

```
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

mach2 = mach*mach
h2 = h*h
eps2 = eps*eps
t1 = 2.d0*dervx(psi,i,j)*(g-1.d0)*mach2
tx = t1*(1.d0-eps2)*exp(-2.d0*psi(i,j))
return
end


function ty(psi,rho,i,j)
real*8 ty,mach,mach2,h,h2,eps,eps2,g
real*8 t1,dervy,psiwall
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

mach2 = mach*mach
h2 = h*h
eps2 = eps*eps
t1 = 2.d0*(g-1.d0)*mach2*(1.d0-eps2)
```

```
ty = t1*dervy(psi,i,j)*exp(-2.d0*psi(i,j))
return
end



function scx(psi,rho,i,j)
real*8 scx,dervx,dervy,rho2,rho3,mach,mach2,h,h2
real*8 eps,eps2,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12
real*8 psiwall,g
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision  psi(n,m),rho(n,m)

mach2 = mach*mach
h2 = h*h
eps2 = eps*eps
rho2 = rho(i,j)*rho(i,j)
rho3 = rho2*rho(i,j)
t1 = rho(i,j)**(g-2.d0)
t2 = (g-1.d0)*t1*dervx(rho,i,j)
t3 = dervx(rho,i,j)*(g-1.d0)*mach2/rho3
t4 = dervx(psi,i,j)*dervx(psi,i,j)
t5 = dervy(psi,i,j)*dervy(psi,i,j)/h2
t6 = -t3*(t4+t5)
t7 = (g-1.d0)*mach2/rho2
```

```
t8 = (psi(i+1,j)-2.d0*psi(i,j)+psi(i-1,j))/(dx*dx)
t9 = (dervx(psi,i,j+1)-dervx(psi,i,j-1))/(2.d0*dy)
t10 = t7*(dervx(psi,i,j)*t8 + t9*dervy(psi,i,j)/h2)
t11 = (g-1.d0)*mach2*(1.d0-eps2)*dervx(psi,i,j)
t12 = -t11*exp(-2.d0*psi(i,j))
scx = t2+t6+t10+t12
return
end



function scy(psi,rho,i,j)
real*8 scy,dervx,dervy,rho2,rho3,mach,mach2
real*8 t1,t2,t3,t4,t5,t6,t7,t8,t9,t10
real*8 psiwall,g,eps,h,h2,t11,t12,eps2
integer i,j
include 'input.h'
common /param/ g,mach,eps,h,psiwall
double precision psi(n,m),rho(n,m)

mach2 = mach*mach
h2 = h*h
eps2 = eps*eps
rho2 = rho(i,j)*rho(i,j)
rho3 = rho2*rho(i,j)
t1 = rho(i,j)**(g-2.d0)
t2 = (g-1.d0)*dervy(rho,i,j)*t1
```

```
      t3 = (g-1.d0)*mach2*dervy(rho,i,j)/rho3
      t4 = dervx(psi,i,j)*dervx(psi,i,j)
      t5 = dervy(psi,i,j)*dervy(psi,i,j)/h2
      t6 = -t3*(t4+t5)
      t7 = (g-1.d0)*mach2/rho2
      t8 = (dervx(psi,i,j+1)-dervx(psi,i,j-1))/(2.d0*dy)
      t9 = (psi(i,j+1)-2.d0*psi(i,j)+psi(i,j-1))/(dy*dy)
      t10 = t7*(dervx(psi,i,j)*t8+dervy(psi,i,j)*t9/h2)
      t11 = -(g-1.d0)*mach2*(1.d0-eps2)*dervy(psi,i,j)
      t12 = t11*exp(-2.d0*psi(i,j))
      scy = t2+t6+t10+t12
      return
      end

*****
***  calculating the change in rho(xi,eta)
*
      function deltrho(psi,rho,dpsi,i,j)
      real*8 dervx,dervy,deltrho,sc,q,r,s,t
      real*8 t1,t2,t3,t4
      integer i,j
      include 'input.h'
      double precision psi(n,m),rho(n,m),dpsi(n,m)

      t1 = r(psi,rho,i,j)*dervx(dpsi,i,j)
      t2 = s(psi,rho,i,j)*dervy(dpsi,i,j)
      t3 = t(psi,rho,i,j)*dpsi(i,j)
      t4 = sc(psi,rho,i,j)
      deltrho = -(t1+t2+t3+t4)/q(psi,rho,i,j)
      return
      end

******
*** the initial guess ...
*
      function psinc(x,y,eps,h)
      real*8 x,y,eps,psinc
      psinc = log(cosh(y/h)+eps*cos(x))
      return
      end

******
*** generic routine for calculating first order x-derivative
*
      function dervx(f,i,j)
      real*8 dervx
      integer i,j
      include 'input.h'
      double precision f(n,m)

      if(i.eq.1.or.i.eq.n) then
      if(i.eq.1) then
      dervx = (-3.d0*f(i,j)+4.d0*f(i+1,j)-f(i+2,j))/(2.d0*dx)
      go to 100
      else
      dervx = (3.d0*f(i,j)-4.d0*f(i-1,j)+f(i-2,j))/(2.d0*dx)
      go to 100
      endif
      else
      endif

      dervx = (f(i+1,j)-f(i-1,j))/(2.d0*dx)
 100  return
      end

******
*** generic routine for calculating first order y-derivative
*
      function dervy(f,i,j)
      real*8 dervy
      integer i,j
      include 'input.h'
      double precision f(n,m)

      if(j.eq.1.or.j.eq.m) then

      if(j.eq.1) then
      dervy=(-3.d0*f(i,j)+4.d0*f(i,j+1)-f(i,j+2))/(2.d0*dy)
      go to 111
      else
      dervy=(3.d0*f(i,j)-4.d0*f(i,j-1)+f(i,j-2))/(2.d0*dy)
      go to 111
      endif
      else
      endif

      dervy = (f(i,j+1)-f(i,j-1))/(2.d0*dy)
 111  return
      end
```

Bibliography

[1] Abramowitz, M. & Stegun, A. 1970 *Handbook of Mathematical Functions.* 9th edn. Dover.

[2] Baker, G.R., Saffman, P.G. & Sheffield, J.S. 1976 Structure of a linear array of hollow vortices of finite cross section. *J. Fluid Mech.* **74**, 469–476.

[3] Bateman, H. 1952 *Partial Differential Equations of Mathematical Physics.* pp. 166-169. Cambridge University Press.

[4] Brown, S.N. 1965 The compressible leading-edge vortex. *J. Fluid Mech.* **22**, 17–32.

[5] Currie, I.G. 1974 Fundamental Mechanics of Fluids. McGraw-Hill Book Company.

[6] Dimotakis, P.E. 1991 Turbulent Free Shear Layer Mixing and Combustion. *Prog. Astro. and Aero.* AIAA **137**, 265-340.

[7] Drazin, P.G. & Reid, W.H. 1981 Hydrodynamic Stability. Cambridge University Press, London.

[8] MALLIER, R. 1994 Stuart Vortices in a stratified mixing layer. I: The Garcia model. *Geophys. Astrophys. Fluid Dynamics.* **74**, 73-97.

[9] GOLUB, G.H. & VAN LOAN, C.F. 1989 Matrix Computations. The Johns Hopkins University Press.

[10] HUANG, M.K. AND CHOW, C.Y. 1982 Trapping of a free-vortex by Joukowski airfoils. *AIAA J.* **20**, 292–298.

[11] KUO, Y.H. & SEARS, W.R. 1954 Plane subsonic and transonic potential flows. In *General Theory of High-Speed Aerodynamics, Vol. VI, High Speed Aerodynamics and Jet propulsion* (ed. W.R. Sears), pp. 490-577. Princeton University Press.

[12] LAMB, H. 1932 Hydrodynamics. Cambridge University Press.

[13] LANDAU, L.D. & LIFSHITZ, E.M. 1959 Fluid Mechanics Pergamon Press.

[14] MACK, L.M. 1960 The compressible viscous heat-conducting vortex.*J. Fluid Mech.* **8**, 284–292.

[15] MALLIER, R. AND MASLOWE, S.A. 1993 A row of counter-rotating vortices. *Phys. Fluids A* **5**, 1074–1075.

[16] MILNE-THOMSON, L. 1966 *Theoretical Aerodynamics* Macmillan.

[17] MOORE, D.W. 1985 The effect of compressibility on the speed of a vortex ring. *Proc. R. Soc. Lond.* A **397**, 87-97.

[18] MOORE, D.W. & PULLIN, D.I. 1987 The compressible vortex pair. *J. Fluid Mech.* **185**, 171–204.

[19] PRESS, W.H. & TEUKOLSKY, S.A. & VETTERLING, W.T. & FLANNERY, B.P. 1992 Numerical Recipes in Fortran Cambridge University Press.

[20] PIERREHUMBERT, R.T. & WIDNALL, S.E. 1982 The two- and three-dimensional instabilities of a spatially periodic shear layer. *J. Fluid Mech.* **114**, 59–82.

[21] RINGLEB, F. 1940 Exakte Losungen der Differentialgleichungen einer adiabatischen Gasstromung *Z.A.M.M.* Bd. 20, Heft4, pp. 185-198. (Available as R.T.P. Translation No. 1609, British Ministry of Aircraft Production.)

[22] ROSHKO, A. 1976 Structure of turbulent shear flows: a new look *AIAA J.* **14**, 1349–1357.

[23] SAFFMAN, P.G. & SHEFFIELD, J.S. 1977 Flow over a Wing with an Attached Free Vortex *Stud. Appl. Maths* **57**, 107–117.

[24] SHAPIRO, A.H. 1953 The Dynamics and Thermodynamics of Compressible Flow. Vol I. John Wiley and Sons.

[25] STUART, J.T. 1967 On finite amplitude oscillations in laminar mixing layers. *J. Fluid Mech.* **29**, 417–440.

[26] THOMPSON, P.A. 1972 Compressible Fluid Dynamics. McGraw-Hill.

[27] VON MISES, R. 1958 Mathematical Theory of Compressible Fluid Flow. Academic Press.