von KARMAN INSTITUTE

FOR FLUID DYNAMICS

Technical Note 175

AN APPLICATION OF CONFORMAL MAPPINGS TO GENERAL GRID GENERATION AND POTENTIAL FLOW PROBLEMS

T.R. Bewley

November 1990

RHODE SAINT GENESE BELGIUM

von Karman Institute for Fluid Dynamics
Chaussée de Waterloo, 72
B–1640 Rhode Saint Genèse – Belgium

Technical Note 175

AN APPLICATION OF CONFORMAL MAPPINGS TO GENERAL GRID GENERATION AND POTENTIAL FLOW PROBLEMS

T.R. Bewley

November 1990

TABLE OF CONTENTS

Abstract ii
List of Figures & Tables iii

1. Introduction 1

2. Simple conformal mapping techniques 3

3. The Schwarz-Christoffel Transformation 5
   3.1. Formulation of the equation 5
   3.2. Applying the Schwarz-Christoffel transformation as a grid generator 6
   3.3. Stretching functions 7
   3.4. Creation of ellipse/hyperbola grids 9

4. Generating potential flow solutions 10
   4.1. The mathematical formulation 10
   4.2. Application to unstable liquid displacement flows in porous media 12
   4.3. Application to channel flows and channel design 13

5. Conclusions 15

References 16

Table 1

Figures

Appendix 1 - Example codes

ABSTRACT

The generation of smooth, orthogonal, body fitted grids is an important first step towards the accurate numerical solution of fluid flows using finite difference methods. To produce a suitable grid for a numerical computation, a powerful grid generation technique employing conformal mappings has been developed, and is summarized in this report.

Once a mapping has been established for a specific geometry, using this method to directly compute two dimensional incompressible potential flow solutions is a straightforward process. This technique is also investigated in the present work.

The generalized code developed in this project has been employed in two research projects at the VKI, indicating the viability of this method for easy adaptation to problems of interest. The first example used the transformation to solve potential flows in channels. The second used the transformation to investigate unstable flows in porous media. The code was also invaluable as a grid generator for a numerical computation by the author for his VKI diploma course project. Extension of this code to other grid generation problems is also exemplified. The variety of these examples should illustrate how to apply the present code to problems of future interest.

LIST OF FIGURES

1a-d – Grids and solution for computation of conical flow over a swept fin

2 – A conformal mapping for unconstricted flow over a step

3a-b – Mapping of a uniform grid using equation 2

4a-b – Mapping of a non-uniform grid using equation 2

5 – A conformal mapping for constricted flow over a step

6 – The Schwartz-Christoffel Transformation

7a-c – Sample airfoil grids produced by SCGRID

8a-c – Sample riblet grids

9 – Mapping of ellipse/hyperbola plane to uniform flow plane

10a-c – Potential flow solution for flow through a sharp 90° corner

11 – Amplification of unstable mode of a porous media flow

12 – Photograph of an unstable liquid displacement flow

13 – Mapping for computation of flow in a given channel

14 – Mapping for computation of a channel shape given one channel wall

15a-b – Channel flow solutions

LIST OF TABLES

1 – Polynomial stretching functions for varying grid concentrations.

1. INTRODUCTION

The suitability of a numerical grid for a finite difference computation is as important a consideration as the computation scheme itself. Accuracy of the scheme is greatly increased if the distance between successive grid points varies smoothly and if the grid lines in the **i=constant** and **j=constant** directions are mutually orthogonal and orthogonal to the boundaries. Much better resolution of regions of high gradients (for example, boundary layers and shocks) is obtained if the grid is concentrated where these regions are expected. Further, the resolution of shock locations can be greatly increased if a grid is chosen such that the grid lines approximately fit the expected shape of the shock. The present method satisfies all of the above criteria quite well for a wide variety of possible geometries.

The idea of the application of a conformal mapping to grid generation problems is a two step process :

1) Produce a simple grid in an intermediate plane and express the grid coordinates as complex numbers. In the present code, two intermediate grids are possible : a simple rectangular grid or a grid formed by a series of confocal ellipses and hyperbolae. Both of these grids are self-orthogonal and orthogonal to the horizontal axis. Stretching functions may be used to concentrate grid lines in any region of interest in this intermediate plane quite easily.

2) Use a specific function analytic (continuously differentiable) on the above domain (called a conformal mapping) to map each complex number above to another complex number. This "mapping" is chosen such that the new set of complex numbers represents a new grid covering the desired domain. The horizontal axis of the intermediate plane maps to the boundary of the body in the final grid, as will be shown below.

Smoothness of the intermediate grid spacing is maintained under the mapping (as much as possible) because the mapping function has smooth derivatives everywhere. Analyticity also implies that two lines which intersect orthogonally will be mapped onto two other lines which intersect orthogonally. Thus, the orthogonality of the intermediate grid is also maintained under the mapping.

Conformal mapping techniques provide incompressible potential flow solutions directly. The mathematical basis for this is outlined in section 4.1. One of the greatest advantages of this technique for solving the Laplace equation is that it can account for infinite boundary conditions quite well. This concept is introduced in chapter 2.

Historical note : The code described in this report was originally developed to produce a grid for the numerical computation of the conical Euler equations in the cross-flow plane of a swept fin [2]. Several algebraic grids were first designed to cover the computational domain in a smooth, orthogonal fashion. They could be made orthogonal (figure 1a) or smoothly varying in size (figure 1b), but never both. The main problem with grids made with such algebraic formulae was that they were only effectively curved in one direction. An example of a much better grid, produced by the code developed in this report, is shown in figure 1c. It was found that, in many cases, the grid constructed via this method fit the shape of the shock quite well.

2. SIMPLE CONFORMAL MAPPING TECHNIQUES

The theory of conformal mappings is quite well developed [1,3,4,5]. Many tables of direct transformation functions are available in the literature [4,5]. As a warm-up to the more general transformation developed in chapter 3, grid generation via two of these functions will now be discussed. This will introduce the properties of stretching and infinite boundary conditions in addition to outlining the general method used.

It can be found in the tables [5] that the transformation :

$$w = \frac{h}{\pi}[(z^2-1)^{1/2} + \cosh^{-1} z] \tag{1}$$

maps the upper half of the z-plane to a domain above a "step" in the w-plane as shown below, where **z=x+iy** and **w=u+iv** are complex numbers representing grid coordinates and **h** is the height of the step.

The convention used for such plots (Fig. 2) is that the hashed region represents the area just outside the domain under consideration.

Via rearrangement of the equation defining the hyperbolic cosine, it is easily seen that equation 1 can be cast in a form that can be solved using the complex function routines available in VAX/VMS :

$$w = \frac{h}{\pi}\{(z^2-1)^{1/2} \; + \; \log[z + (z-1)^{1/2}]\} \tag{2}$$

The first step in creating a grid over the step in the w-plane is creating a grid in the ż-plane. Let us choose a simple uniform rectangular grid, and map each point to its new location using the above formula.

Both grids shown in figure 3 happen to form a set of streamlines and potential lines for incompressible potential flow. The mathematical explanation for this will be postponed to section 4.1.

The code required to produce figure 3 is fairly simple, and is given in the appendix.

The resolution of the regions near the corners can be improved if artificial "stretching" is employed. Both grids will still represent streamlines and potential lines, but now with non-uniform spacing. To do this, it would be useful to be able to prescribe the grid points on the lower boundary of the grid directly in the w-plane, and let the grid "grow" away from these points. However, the grid still needs to be produced first as a rectangular grid

in the z-plane. Therefore, the prescribed points in the w-plane must first be mapped back to the z-plane. To do this, the inverse mapping z=g(w) is required. However, equation 2 is not easily solved for z. This problem can be sidestepped : it is known that the image of any of these prescribed points will lie on the x-axis of the z-plane, somewhere around the origin. Using a numerical "root false position" scheme [6], the value of x as a function of w may by found to a very high degree of accuracy quite quickly. Once this is done, an intermediate rectangular grid terminating at these points is easily produced, and then mapped to the w-plane just as before, (see figure 4).

Now consider the mapping [5] :

$$w = \cosh^{-1}\frac{2z-h-1}{h-1} - \frac{1}{h}\cosh^{-1}\frac{(h+1)z-2h}{(h-1)z} \tag{3}$$

which maps the half the upper half of the z-plane to a restricted domain above a step in the w-plane as shown in figure 5.

The potential lines in the w-plane connecting E'-D' and F'-A' map back to curves in the z-plane which connect E-D an F-A. It is known that a point source solution (a set of circles and rays) satisfies this requirement. The potential for a point source may thus be chosen as the intermediate grid, with grid lines which happen to land at points B and C in the z-plane. These grid points are then mapped via equation 3, and the solution is again easily found. The details of this mapping are left to the reader.

Both of the mappings illustrate the outstanding quality of conformal methods to simulate infinite boundary conditions. *Note that the streamlines and potential lines on the boundaries corresponding to $x > 0$ are correctly curved.* The grid need not extend in the horizontal and vertical directions a distance large enough compared to the size of the step that "uniform" conditions can be approximated. Further, accuracy of the solution is independent of grid spacing. In fact, the entire grid could be removed and only those points in the region of interest (for instance, on the boundary), could be treated. These qualities set conformal methods in a class apart from standard Laplace solvers.

3. THE SCHWARZ-CHRISTOFFEL TRANSFORMATION

3.1. Formulation of the equation

The differential form of the Schwarz-Christoffel Transformation is given by :

$$\frac{dz}{d\zeta} = f'(\zeta) = M \prod_{i=1}^{n} (\zeta - a_i)^{\alpha_i/\pi} \tag{4}$$

This formula represents a conformal mapping of the upper half of the ζ-plane onto a region in the z-plane bounded by a straight line with a finite number of corners. The points a_i on the real axis in the ζ-plane map to corners with an outside turning angle α_i on the edge of the domain in the z-plane. This is illustrated in figure 6.

The α_i are the outside angles of the corners, defined positive for clockwise rotation when the curve is followed in the direction of the enumeration of the corners, where the enumeration of the a_i in the ζ-plane is taken from left to right. Thus, the α_i in figure 6 are all negative. If the sum of the α_i is equal to -2π, the half plane is mapped to the interior of a closed polygon; if the sum is equal to 2π, the half plane is mapped to the exterior of a closed polygon. It is also possible to map to more general shapes if the sum of the α_i is between $-\pi$ and π. For example, the code developed for this project was originally written to map onto the geometry shown in figure 1c, for which the sum of the α_i is $-\pi/2$.

A difficulty arises in applying the transformation $z=f(\zeta)$ because the integral of equation 4 cannot, in general, be evaluated directly. A second order accurate integration formula can be found by extending the trapezoid rule to each of the terms separately, as done by Davis [1], leading to the following form :

$$z_{m+1} - z_m = \frac{M}{(\Delta\zeta)^{n-1}} \prod_{i=1}^{n} \left[\left(\frac{(\zeta - a_i)^{(\alpha_i/\pi)+1}}{(\alpha_i/\pi)+1} \right)_{\zeta_m}^{\zeta_{m+1}} \right] \tag{5}$$

It was found that this integration formula works quite well for most applications. In the vicinity of very sharp corners for which α_i is less than $-\pi/2$, one must use some caution that the step size used is sufficiently small for the algorithm to be accurate.

A difficulty arises in applying equation 5 because the constants a_i (the location of the images of the corners on the ξ-axis) and M (a complex constant which rotates and scales the mapping) are not known *a priori*. As it turns out, only two of the a_i may be set arbitrarily. Note that one of these may be chosen to be $a_i=\infty$, in which case the influence of this corner in the transformation vanishes. The rest of the a_i and M have to be computed iteratively so that the transformation maps the x-axis in the ζ-plane to the

desired shape at the desired location with the desired rotation and scaling in the z-plane. An algorithm to find the remaining constants has been outlined by Davies [1]. Given the desired corner locations $z_{true,i}$, this algorithm was implemented to find the a_i and M in the following manner :

A. Assign values for all the $a_{guess,i}$, with one of the corners at $a_{guess,i1}$=-1, one of the corners at $a_{guess,i2}$=1, and the other $a_{guess,i}$ in the same order on the ξ-axis as the corners are enumerated on the body in the z-plane, as in figure 6.

B. Given these assumed values of the $a_{guess,i}$ and the true location in the z-plane of the corners i1 and i2, compute the (complex) value of M by equation 5.

C. Starting at i=i1 with this value of M and the assumed values $a_{guess,i}$, compute the locations in the z-plane of the corners corresponding to the i≠i1, i2 by equation 5.' Call these values $z_{guess,i}$. Note that $z_{guess,i1} \equiv z_{true,i1}$ and $z_{guess,i2} \equiv z_{true,i2}$.

D. Starting at i=i1 with $a_{new,i1}$=-1 and K=1, compute values of the $a_{new,i}$ for i≠i1 by :

$$\frac{a_{new,i} - a_{new,i-1}}{a_{guess,i} - a_{guess,i-1}} = K \left| \frac{z_{true,i} - z_{true,i-1}}{z_{guess,i} - z_{guess,i-1}} \right| \tag{6}$$

E. Compute the value of K which gives $a_{new,i2}$=1.

F. Recompute the $a_{new,i}$ with the corrected value of K in equation 6, rename these values $a_{guess,i}$, and go back to step B.

Iterate until the $z_{guess,i}$ are sufficiently close to the $z_{true,i}$. In the present code, convergence is achieved to 4 or 5 place accuracy for the location of each of the corners in the z-plane, and thus this accuracy everywhere in the domain, usually within 15 iterations. Accuracy of the initial guesses of the a_i to their final values is not critical; any reasonable spacing converges quickly.

3.2. Applying the Schwartz-Christoffel transformation as a grid generator

The conformal mapping outlined in the previous section may now be applied as a grid generator in a method identical to that described in the chapter 2. However, this new conformal mapping may be applied to very general shapes with arbitrary numbers of corners. Curves can be handled simply by approximation with a finite number of corners. (Curves can also be handled directly by the Schwartz-Christoffel transformation, as pointed out by Davis. This possibility is not discussed here, but would be a very interesting extension of this project). The extended range of applications is what makes

this mapping so attractive.

To illustrate the flexibility of this grid generation technique, a code called SCGRID was developed and generalized. A series of grids produced by this code are presented in figure 7 to illustrate its capabilities.

The present transformation maps the horizontal axis onto a prescribed curve. Thus, if one wishes to impose the shape of one of the edges of the grid, a rectangular grid (with one edge terminating on this axis) is chosen as the intermediate grid. Figure 7c was produced using this method. If one wishes to impose the shape of three of the edges of the grid, a ellipse/hyperbola grid (with three edges terminating on this axis) is chosen as the intermediate grid. Figures 7a, 7b, and 8b were produced using this method.

3.3. Stretching functions

Stretching functions are used by SCGRID to concentrate the grid in certain regions. For instance, for a viscous computation, high grid density is desired near the wall to resolve the boundary layer; for an inviscid computation, a lower grid density near the wall is allowed.

In order to provide grid density in the desired areas using the technique of chapter 2, an inverse mapping might be useful. However, the inverse mapping of equation 4 is unattainable, as were the inverse mappings of equations 1 and 3. Since it is known that the boundary in the z-plane must map back to the ξ-axis, a root false position scheme (or any other root solver in one variable) may again be used for the inverse mapping of the boundary points. However, SCGRID avoids doing this at all. With a little ingenuity, it is always possible to apply stretching in the intermediate plane directly to give the desired results in the final grid. It is actually preferable to do this; the resulting grid spacing is generally found to be smoother using this technique.

In each case that a stretching function is needed by SCGRID, an index is first created such that **index=0** corresponds to the first point and **index=1** corresponds to the last. A polynomial is then used to space the grid points from one extreme to the other (exactly how this is done will be shown below). This polynomial is used to concentrate grid points in varying degrees near each extreme.

One way to concentrate grid points at an extreme is to make the stretching function flat there by setting one or more derivatives of the polynomial equal to zero at the extreme and then solving the resulting system of equations for the coefficients of the desired polynomial. The polynomial is of an order equal to the number of conditions imposed minus 1. Table

1 is a list of such polynomial stretching functions that may be useful.

Note that it is not really necessary to set a derivative equal to zero to change the stretching. More subtle degrees of stretching may be obtained by constructing different polynomials. It was found for the application described in chapter 1 that a stretching function of $\mathbf{f(x)=-x^3+2x^2}$ (half way between $\mathbf{f(x)=x^2}$ and $\mathbf{f(x)=-2\ x^3\ +\ 3x^2}$) provided just a little grid concentration near the boundary **i=II** (the free stream), and was quite useful for capturing the shock near this boundary in a very narrow region.

As an example, consider the stretching function required to generate a series of **II** ellipses, whose **y**-intercepts vary from **y=0** to **y=LL**. For each ellipse, an index is normalized according to :

$$\text{index(i)} = \frac{i-1}{\pi - 1} \quad , \quad (7)$$

Note that **index(1)=0** and **index(II)=1**. The **y**-intercepts of the ellipses are now distributed according to :

$$\text{y(index(i))} = \text{LL} * \text{f(index(i))} \quad (8)$$

where **f(index)** is an appropriate polynomial stretching function chosen from table 1. For a viscous computation, the stretching function f(index)=index^2 provides a good grid density near the wall (index=0) necessary for resolving the boundary layer.

Similarly, stretching functions may be used to space the **x**-intercepts of a series of **JJ** hyperbolae between **x=-1** and **x=1**. Stretching functions may also be used to concentrate rectangular grids near a line **x=constant** or **y=constant.**

Many other stretching functions may be employed. It is found that a hyperbolic tangent function is a stretching function well suited for computing a boundary layer because (when properly chosen) the density of the grid can be made to be approximately proportional to the gradients of the velocity present in the boundary layer plus a constant corresponding to the grid density in the (uniform) freestream. For those flows which are predominantly boundary layer problems, the hyperbolic tangent may also be used by SCGRID as another type of stretching away from the wall. However, the flexibility given by polynomial stretching functions (listed above, or others of the user's creation, with grid concentration elsewhere) have proven to be quite beneficial for the creation of appropriate grids over the difficult geometries that this program was designed to accommodate.

3.4. Creation of ellipse/hyperbola grids

The expressions for a series of ellipses and hyperbolae with common focal points e_1=-1 and e_2=1 are :

Ellipse :	$\frac{x^2}{(1+b^2)} + \frac{y^2}{b^2} = 1$	where b=y-intercept of ellipse	(9a)
Hyperbola :	$\frac{x^2}{a^2} - \frac{y^2}{(1-a^2)} = 1$	where a=x-intercept of hyperbola	(9b)

This system of equations may be solved for x and y. Thus, for a given ellipse (b) and a given hyperbola (a), the location of the intersection point is easily found. Note that, by multiplying through by b^2 and $(1-a^2)$ respectively, the above equations hold for b=0 and for a=1, i.e., they hold up to and including the points which lie on the x-axis.

It has therefore been shown that, after stretching functions are applied to compute a series of a_i and b_i, an ellipse/hyperbola grid may be generated from these intercept points directly.

4. GENERATING POTENTIAL FLOW SOLUTIONS

4.1. The mathematical formulation

The steady, incompressible, inviscid, irrotational flow equations may be cast as simply the Laplace equation for the velocity potential :

$$\frac{\partial^2\varphi}{\partial\xi^2} + \frac{\partial^2\varphi}{\partial\eta^2} = 0 \qquad (10a)$$

This equation is linear. Therefore, the equation can be mapped via a conformal mapping to give the Laplace equation in a new system of coordinates :

$$\frac{\partial^2\varphi}{\partial x^2} + \frac{\partial^2\varphi}{\partial y^2} = 0 \qquad (10b)$$

Thus, if a solution to Laplace's equation φ is mapped conformally from the ζ-plane to the z-plane, the same values of φ on the new domain will also satisfy Laplace's equation in the new coordinate system.

The steady compressible potential equation, however, is highly nonlinear :

$$\frac{\partial^2\varphi}{\partial x^2}(u^2 - a^2) + \frac{\partial^2\varphi}{\partial y^2}(v^2 - a^2) + 2uv\frac{\partial^2\varphi}{\partial x\partial y} = 0 \qquad (11)$$

Solutions of this equation do not map conformally to solutions in new coordinate systems due to the non-linearity of the equation.

Viscous solutions and rotational solutions also cannot be mapped. Since there is no velocity potential for such flows, the components of the velocity themselves would have to be mapped. However, the momentum equation is clearly non-linear with respect to the components of velocity :

$$\frac{\partial}{\partial t}\rho u_i = -\frac{\partial}{\partial x_k}\rho u_i u_i + \rho\chi_i - \frac{\partial\tau_{ik}}{\partial x_k} \qquad (12)$$

Thus, the application of conformal mappings as a direct solver for fluid flows is limited to incompressible, irrotational (potential) flows. Note that the addition of unsteady behaviour does not alter the incompressible potential equation - unsteadiness is simply applied as a time dependent boundary condition. Thus, unsteady flows may also be considered by this technique.

One simple solution to the Laplace equation is uniform flow. As shown previously, the Schwartz-Christoffel transformation maps the upper half plane to a region bounded by a surface with a finite number of corners. One such mapping is onto a strip.

Under the transformation shown in Fig. 9, a series of confocal ellipses and hyperbolae in the ζ-plane (with foci at ± 1) maps to a series of horizontal and vertical lines in the z-plane.

Now, a series of horizontal and vertical lines in the z-plane can be considered a trivial solution of Laplace's equation. Thus, by the discussion given in the previous section, a series of ellipses and hyperbolae in the ζ-plane are also a solution of Laplace's equation (for some particular set of boundary conditions).

These two solutions of Laplace's equation (rectangular grids and ellipse/hyperbola grids) are used as intermediate grids by the present code. Thus, all grids produced via this code are possible potential flow solutions, given that they happen to satisfy the correct boundary conditions for the final grid. Since the grids are always normal to the boundaries, this is found to be the case quite often.

In the present technique for solving potential flows, a straight-forward method is employed. First, each grid point where the solution is to be found is considered in the "uniform flow" plane. This is done via the conformal mapping used to produce the grid itself (the intermediate grid points are actually already known) and, if the intermediate plane was the "ellipse-hyperbola" type, the half-plane onto the strip mapping shown in Fig. 9. This point is then incremented some small amount $\Delta\phi$ in this uniform flow plane, in a direction taken to be the direction of the flow in this plane (up, down, left, or right). This new point is then mapped back to the "solution" plane and compared to the original point.

The change in potential between these two points in the "solution" plane is exactly the same as the change in potential between their image points in the "uniform flow" plane, $\Delta\phi$, by the discussion in the previous section. Thus, the velocity in the "solution" plane is in the direction of the new point with a magnitude of $u=\Delta\phi/\Delta s$, where Δs is the distance between the two points in the solution plane. The velocities on the entire field are then scaled by whatever constant desired, density is taken to be constant, and the pressure is computed via Bernoulli's equation on the solution plane compared to a reference point somewhere in the domain.

Although the Schwartz-Christoffel transformation is indeed analytic everywhere in the domain, it is NOT analytic on the corner points of the boundary. Indeed, equation

4 shows that the derivative makes a jump at these points. At corners which turn "into" the flow (inside corners), this non-analyticity does not present a problem for computing potential flow solutions - the velocity goes to zero near such corners as expected. At corners which turn "away" from the flow (outside corners), the computed velocity tends toward infinity near these corners as the distance Δs gets very small. This means that the pressure (computed via Bernoulli's equation) must go negative somewhere in the vicinity of such points - clearly a non-physical result. This behaviour is indeed to be expected; potential flow solutions do blow up at such outside corners. As a point of fact, incompressible viscous flow solutions also blow up in the vicinity of sharp outside corners. Only when compressibility effects are considered can such regions be handled correctly arbitrarily close to the corner. Thus, the divergence of the solution near outside corners is a non-physical result of the equation itself and not a drawback of the solution method employed.

A simple potential flow solution illustrating this property is the flow through a sharp 90° corner. This solution is shown in figure 10.

4.2. Application to unstable liquid displacement flows in porous media

Flows through porous media are governed by the following equations [7] :

modified equation of continuity	$\epsilon \frac{\partial \rho}{\partial t} = -(\nabla \cdot \rho v_o)$	(13a)
Darcy's law :	$v_o = -\frac{\kappa}{\mu}(\nabla p - \rho g)$	(13b)
equation of state :	$\rho = \rho_o p^m e^{\beta p}$	(13c)

in which ϵ is the porosity (ratio of pore volume to total volume), $\mathbf{k}$ is the permeability, and $\mathbf{v_0}$ is the superficial velocity (volume rate of flow through a unit cross-sectional area of the solid plus fluid) averaged over a small region of space - small with respect to the macroscopic dimension of the flow but large with respect to the pore size. In the case of incompressible fluids, neglecting the effect of gravity, these equations may be reduced to :

$$\Delta p = 0 \tag{14a}$$

$$v_0 = -\frac{\kappa}{\mu}\nabla p \tag{14b}$$

The conformal mapping method previously developed has been shown to be an efficient solver of Laplace's equation for the velocity potential. In this type of porous media flow, the velocity potential field is simply the pressure field. Thus, this problem may be treated by the present technique.

A specific porous media flow of particular interest is the unstable forced injection of a low viscosity fluid into a high viscosity fluid ($\mu_1/\mu_2 \approx 1000$). This is a problem of a high degree of interest, for it models a method of obtaining oil from an underground deposit via the injection of seawater.

Such flows may be modeled with the equations listed above, with some constant pressure difference between the surface of the injection "bubble" and infinity (in the high viscosity fluid). It is soon realized that any small perturbation on the surface of the bubble tends to amplify, thus resulting in an instability. Deposits of the high viscosity fluid are left trapped in the valleys as the peaks of the low viscosity fluid grow. These results in a lower yield of oil in the above example, a highly undesirable effect which can be minimized by a closer matching of the viscosities of the two fluids (for example, by the addition of polymers to the seawater). An example of the growth of one Fourier mode of perturbation on a circular bubble is shown below. The inner curve represents $t=t_1$ and the outer curve represents $t=t_6$.

The solution method employed to produce Fig. 11 clearly reveals the strength of the current method. The Schwartz-Christoffel transformation was solved at each time step, then used to compute the velocity of each boundary point. Each boundary point was then stepped according to its individual velocity, and the method repeated. The beauty of the method is that the infinite domain outside the boundary was treated exactly during the computation without ever being actually computed - *the Schwartz-Chritoffel transformation needed only to be considered on the boundary points themselves.*

A photograph of an unstable liquid displacement flow between two closely spaced plates is given Fig. 12, illustrating the same effect. Note that in this flow a certain length scale of perturbation (governed by the distance between the two plates) is amplified most - thus when the boundary becomes longer, higher modes of instability begin to take effect, and the snowflake pattern develops. These effects of preferential length scales of instability due to the surface tension of the fluid are neglected in the current analysis.

4.3. Application to channel flows and channel design

The problem of potential flow in a channel is less suited to the current method of analysis than the problems described previously. This is so because equation 4 cannot formulated as equation 3 was with a domain extending to upstream and downstream infinity. Thus, uniform inlet or outlet conditions must be approximated to compute the flow in a channel given in Fig. 13.

Another application better suited to the present code is channel design. Given one curve of the channel and uniform flow at the inlet and outlet, it is possible to compute the other curve of the channel so it will exactly coincide with a streamline which will turn the flow the required amount. This is illustrated in Fig. 14.

A channel designed to minimize effects of separation was analysed by the above two techniques. The results are shown in Figs. 15a,b.

Note that these two channels do not coincide. This illustrates the point that the given channel may not be the most optimum design from the standpoint that it doesn't quite turn the flow exactly at the outlet of the corner, but the variations extend into the inlet and outlet a short distance. Note that this property could not have been discovered using a standard Laplace solver. The narrower corner designed by the present method turns the flow completely at the outlet and may indeed be a better design for certain applications.

5. CONCLUSIONS

A generalized grid generator/incompressible potential flow solver has been presented. It has illustrated a capability of handling boundary conditions that makes it better suited for the solution of many potential flows than standard Laplace solvers. Many examples of the possible applications of this code have been included. The mathematical formulation of the transformation and its treatment of linear equations are also discussed.

REFERENCES

1. Davis, R.T.: Numerical methods for coordinate generation based on Schwarz-Christoffel Transformations. *AIAA Paper 79-1463*, 1979.

2. Bewley, T.R.: Swept fin induced shock wave/laminar boundary layer interactions - an analytical, computational, and experimental study. *VKI Project Report 1990-34*, June 1990.

3. Carrier, G.F.; Krook, M.; Pearson, C.E.: Functions of a complex variable. *Hod Books, Ithaca, New York*, 1983.

4. Spiegel, M.R.: Theory and problems of complex variables with an introduction to conformal mapping and its applications. *Schaum's Outline Series, McGraw-Hill,, Inc., New York*, 1964.

5. Churchill, R.V. & Brown, J.W.: Complex variables and applications. *McGraw-Hill, New York*, 1984.

6. Press, W.H.; Flannery, B.P.; Teukolsky, S.A.; Vetterling, W.T.: Numerical recipes, the art of scientific computing. *Cambridge University Press*, Cambridge, 1986, pp 498-506.

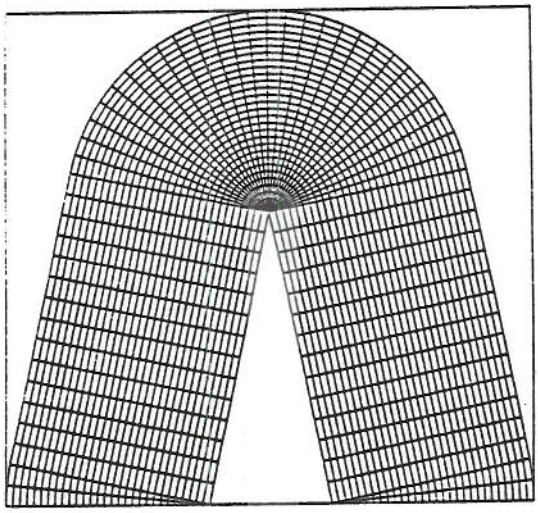
7. Bird, R.B.; Stewart, W.E.; Lightfoot, E.N.: Transport phenomena. *John Wiley and Sons, New York*, 1960.

increasing concentration near index = 0 ---->

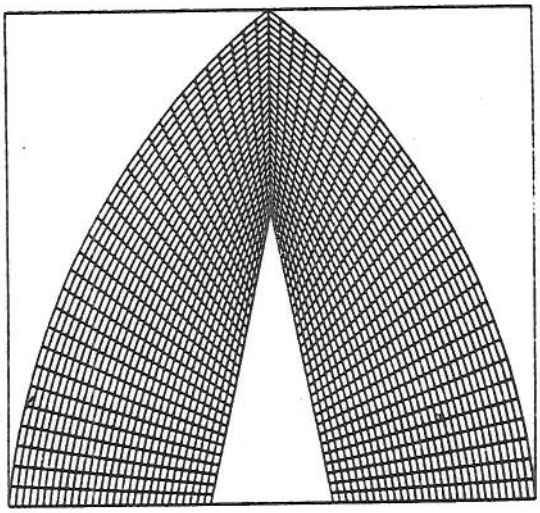
<---- increasing concentration near index = 1

conditions → ↓	$f(0) = 0$	$f(0) = 0$ $f'(0) = 0$	$f(0) = 0$ $f'(0) = 0$ $f''(0) = 0$	$f(0) = 0$ $f'(0) = 0$ $f''(0) = 0$ $f'''(0) = 0$
$f(1) = 1$	$f(x) = x$	$f(x) = x^2$	$f(x) = x^3$	$f(x) = x^4$
$f(1) = 1$ $f'(1) = 0$	$f(x) = -x^2 + 2x$	$f(x) = -2x^3 + 3x^2$	$f(x) = -3x^4 + 4x^3$	$f(x) = -4x^5 + 5x^4$
$f(1) = 1$ $f'(1) = 0$ $f''(1) = 0$	$f(x) = x^3 - 3x^2 + 3x$	$f(x) = 3x^4 - 8x^3 + 6x^2$	$f(x) = 6x^5 - 15x^4 + 10x^3$	$f(x) = 10x^6 - 24x^5 + 15x^4$
$f(1) = 1$ $f'(1) = 0$ $f''(1) = 0$ $f'''(1) = 0$	$f(x) = -x^4 + 4x^3 - 6x^2 + 4x$	$f(x) = -4x^5 + 15x^4 - 20x^3 + 10x^2$	$f(x) = -10x^6 + 36x^5 - 45x^4 + 20x^3$	$f(x) = -20x^7 + 70x^6 - 84x^5 + 35x^4$

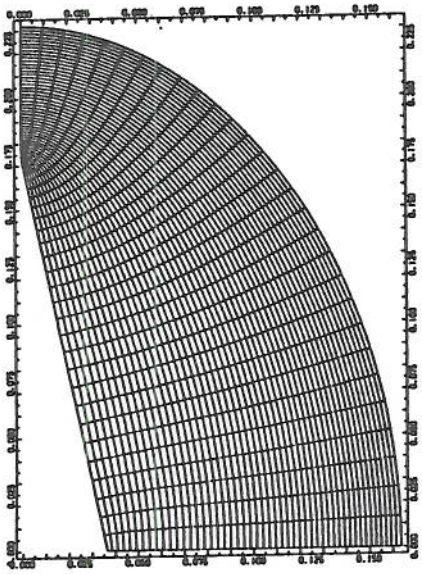
Table 1 – Polynomial stretching functions for varying grid concentrations



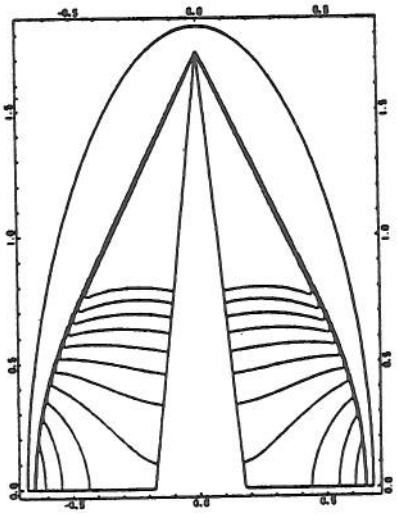
a. – Orthogonal algebraic grid



b. – Smooth algebraic grid



c. – Grid produced via conformal mapping



d. – Solution of conical flow

Fig. 1 – Grids and solution for computation of conical flow over a swept fin

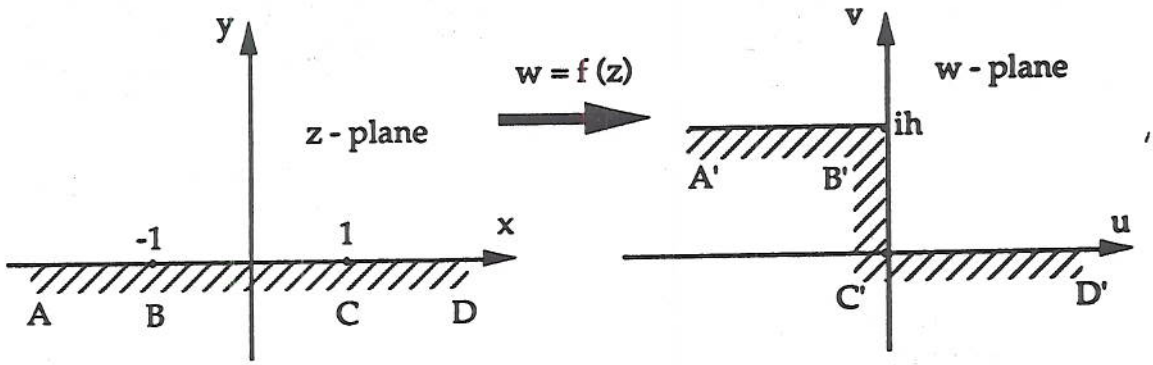


Fig. 2 – A conformal mapping for unconstricted flow over a step

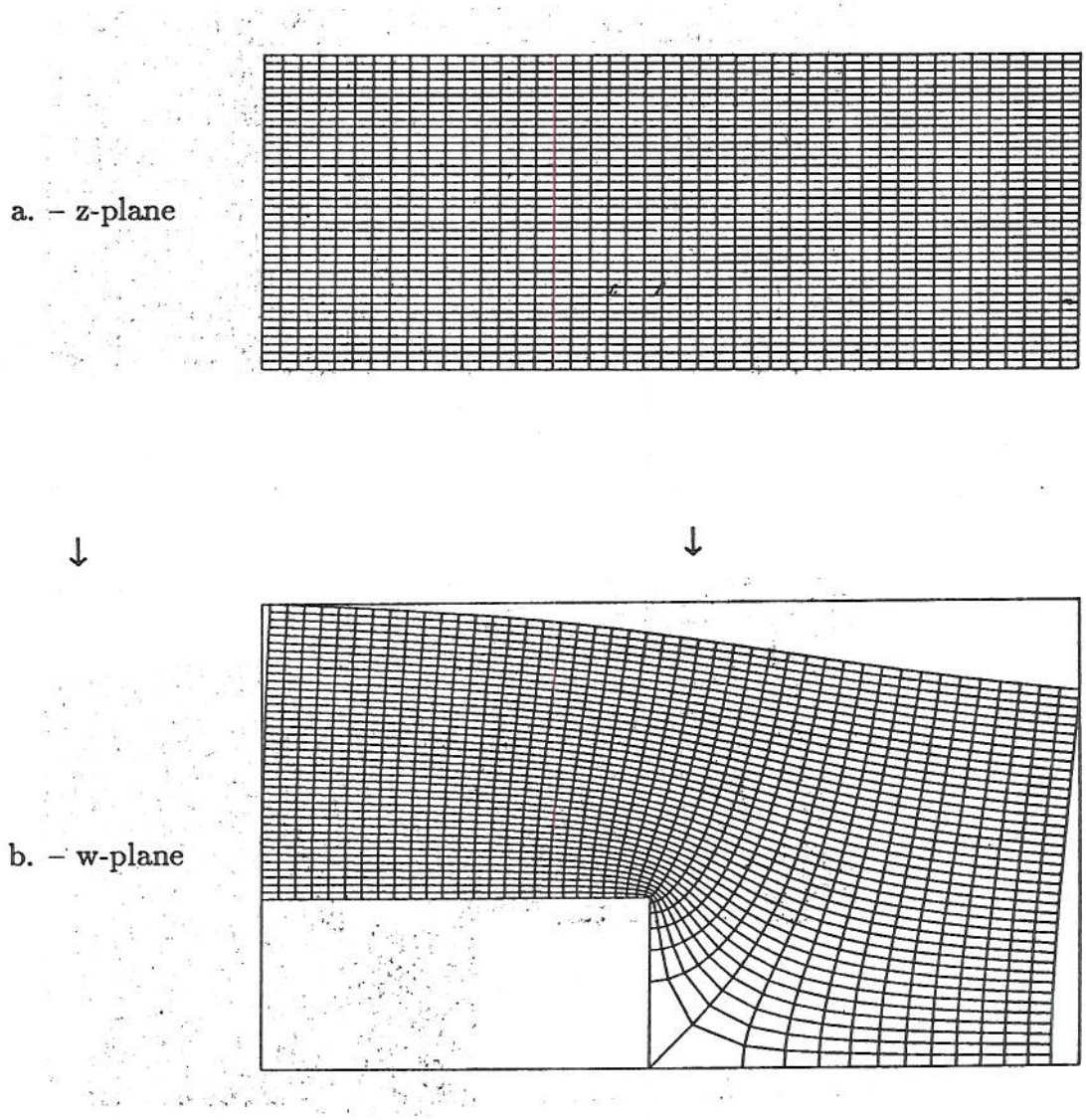
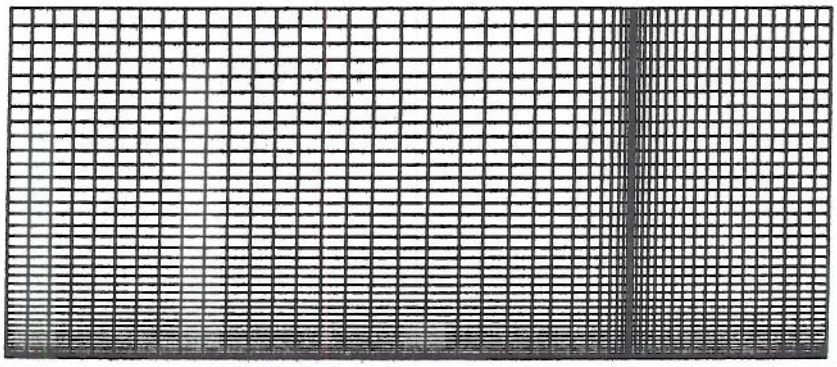


Fig. 3 – Mapping of a uniform grid using equation 2

a. – z-plane



b. – w-plane

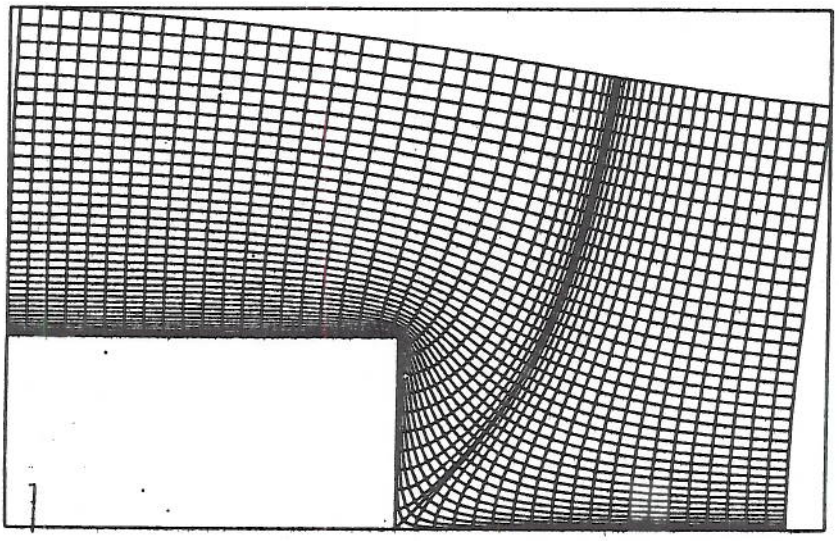


Fig. 4 – Mapping of a non-uniform grid using equation 2

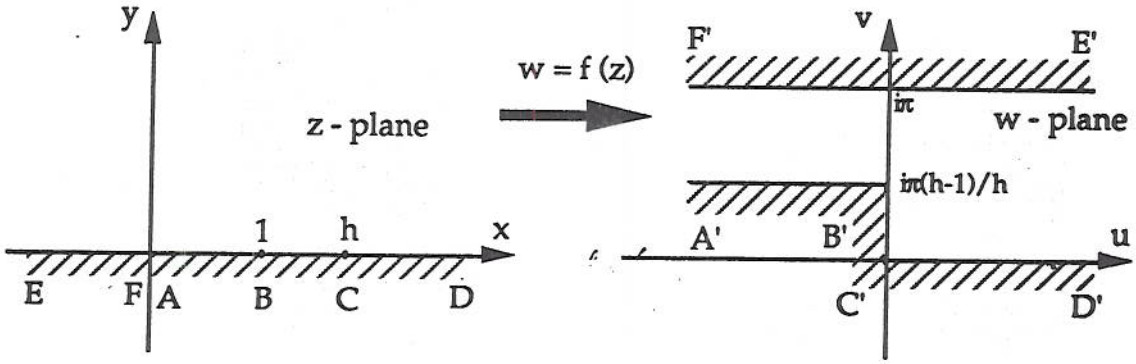


Fig. 5 – A conformal mapping for constricted flow over a step

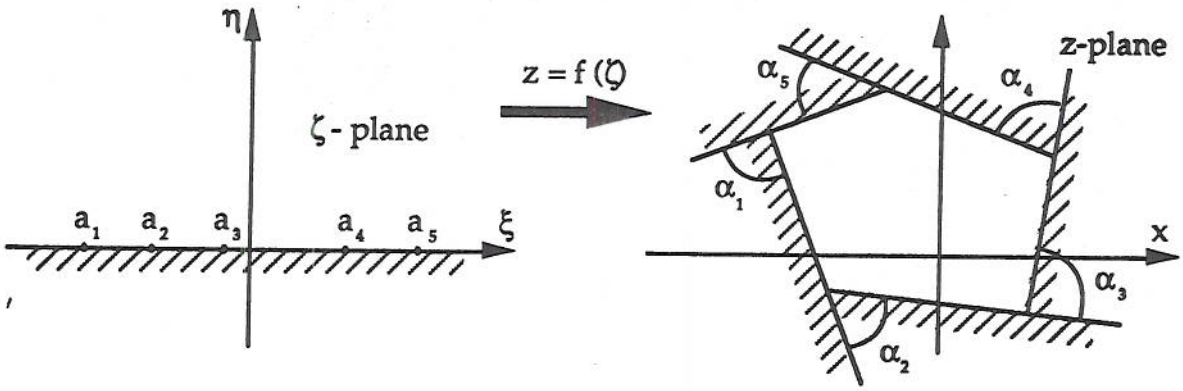
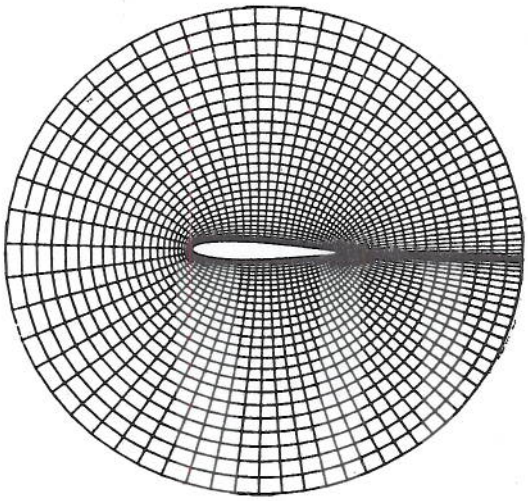
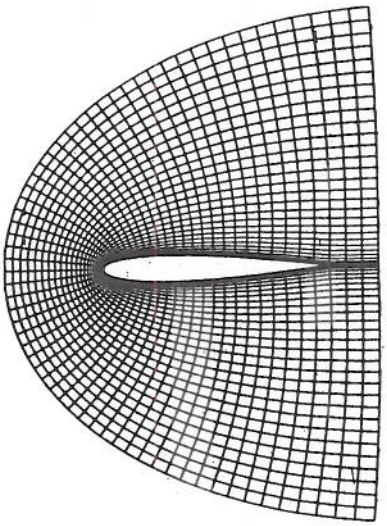


Fig. 6 – The Schwarz-Chritoffel Transformation

a. – Circular grid



b. – C-grid



c. – Potential flow grid

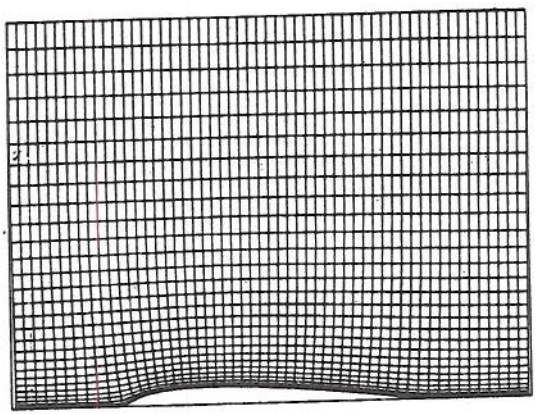
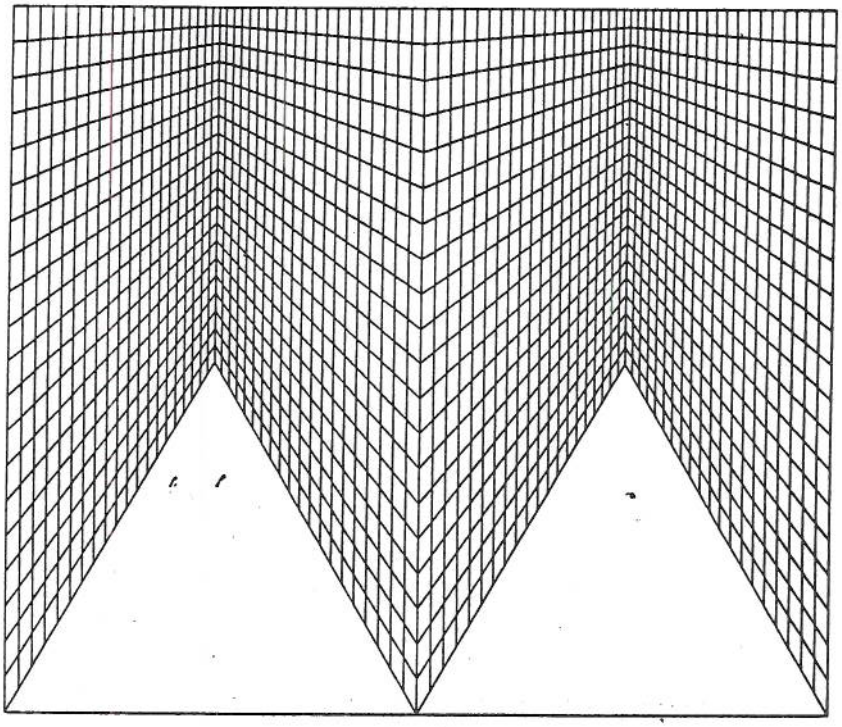
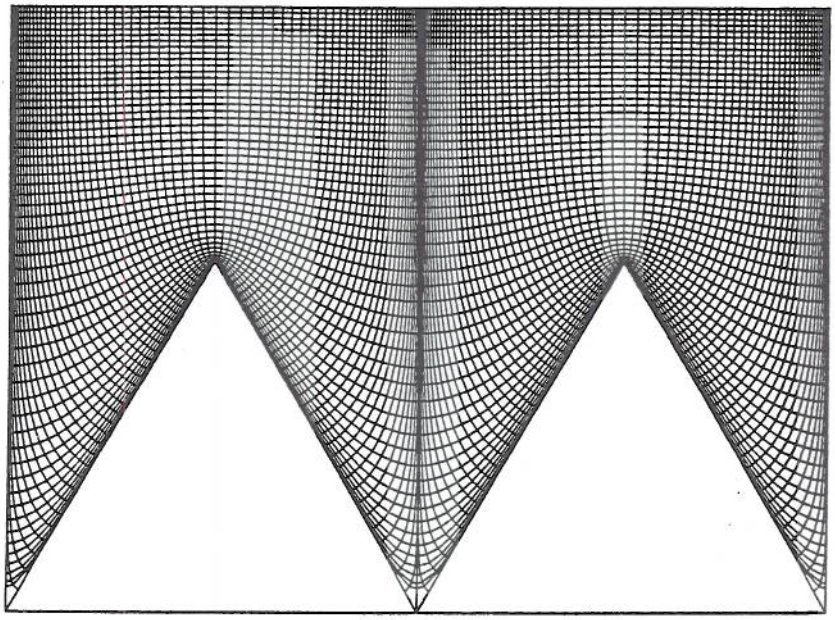


Fig. 7 – Sample airfoil grids produced by SCGRID

a. – Algebraic riblet grid



b. – Riblet grid produced by SCGRID



c. – Intermediate grid for figure 8b

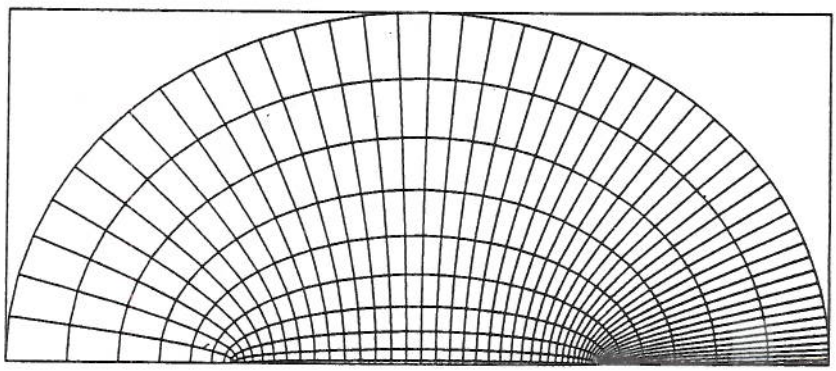


Fig. 8 – Sample riblet grids
(Note that Fig. 8b is exactly a set of potential lines and streamlines for flow over the riblet)

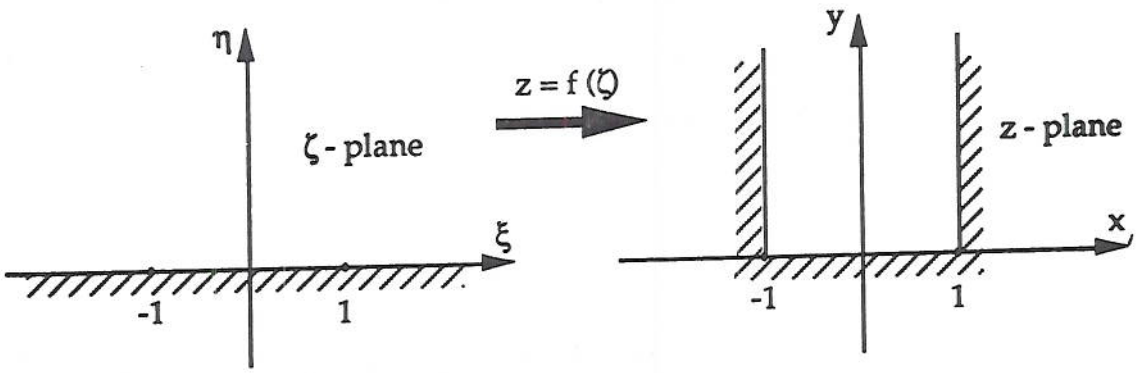
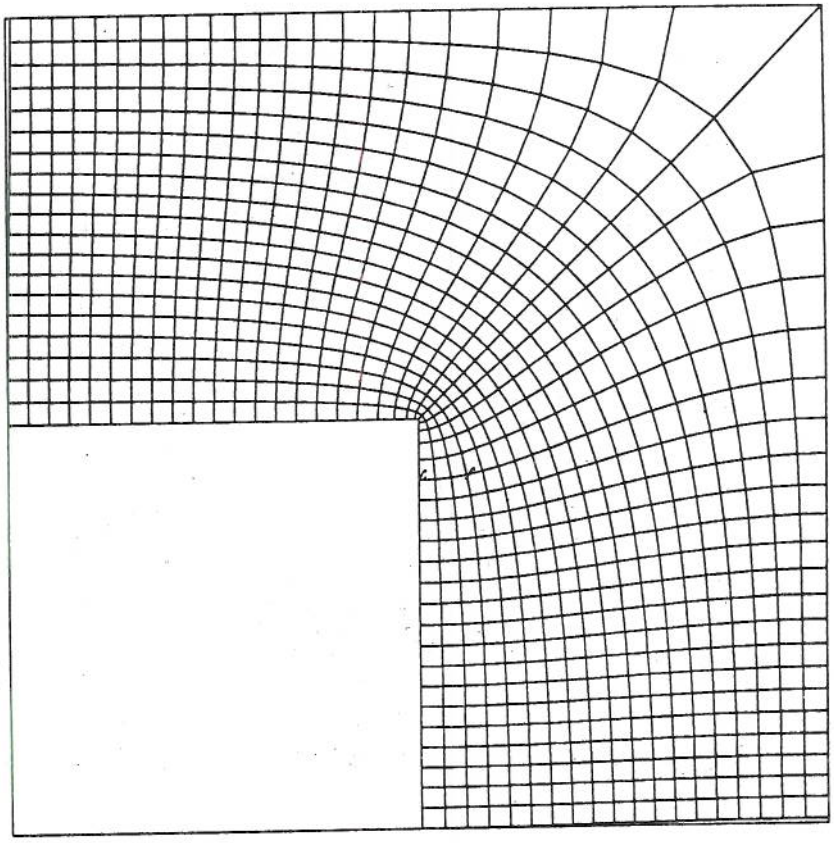
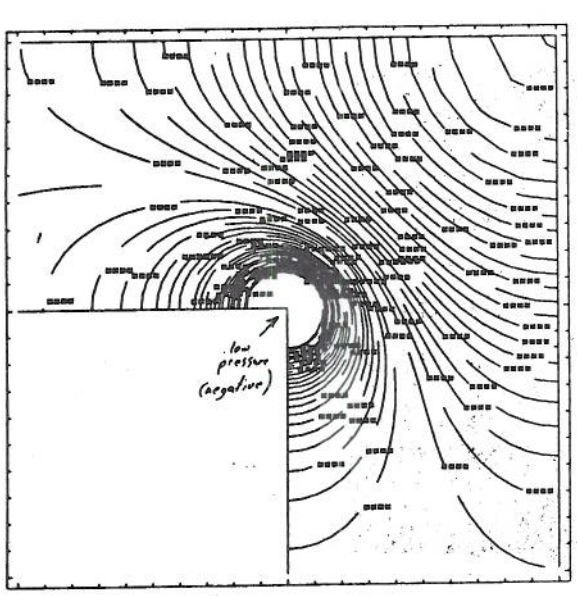


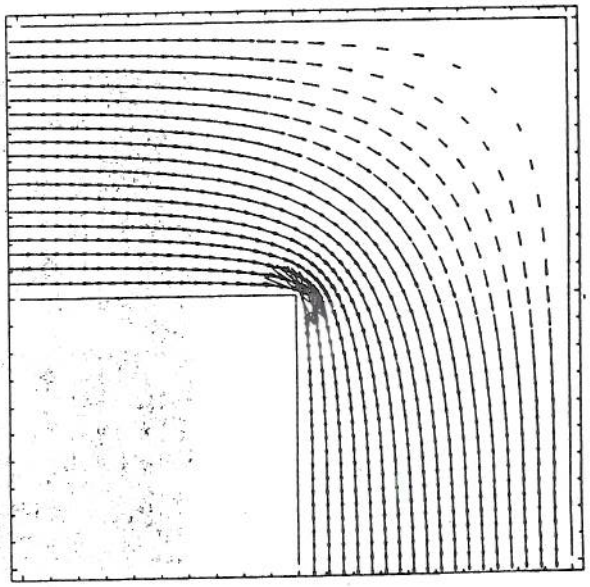
Fig. 9 – Mapping of ellipse/hyperbola plane to uniform flow plane



a. – Streamlines and potential lines



b. – Pressure contours



c. – Velocity vectors

Fig. 10 – Potential flow solution for flow through a sharp 90° corner

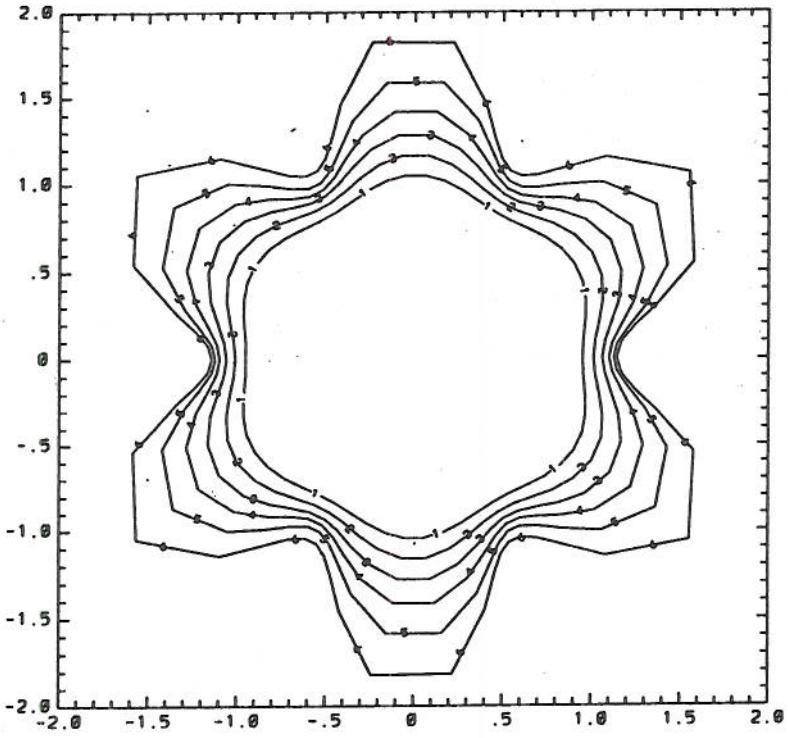


Fig. 11 – Amplification of unstable mode of a porous media flow

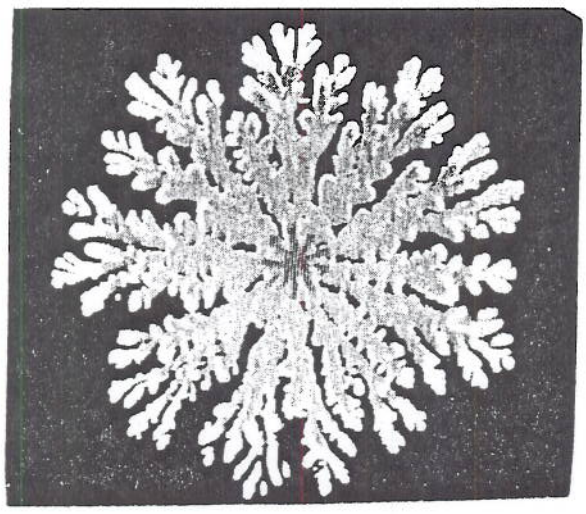


Fig. 12 – Photograph of an unstable liquid displacement flow

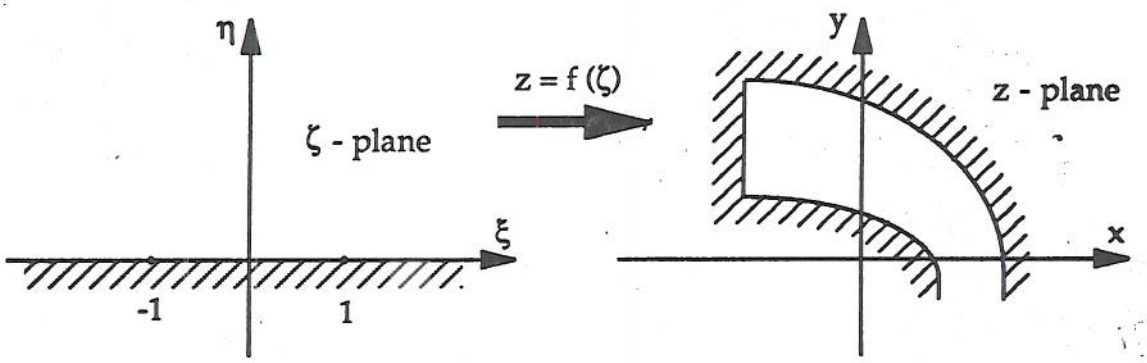


Fig. 13 – Mapping for computation of flow in a given channel

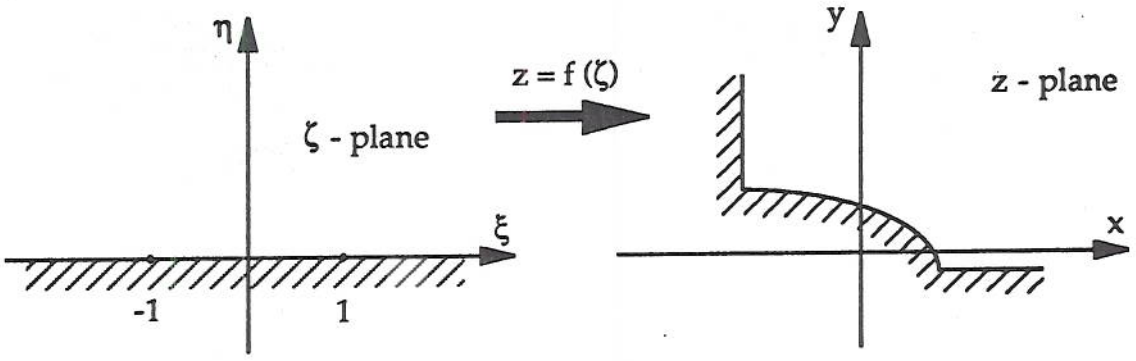
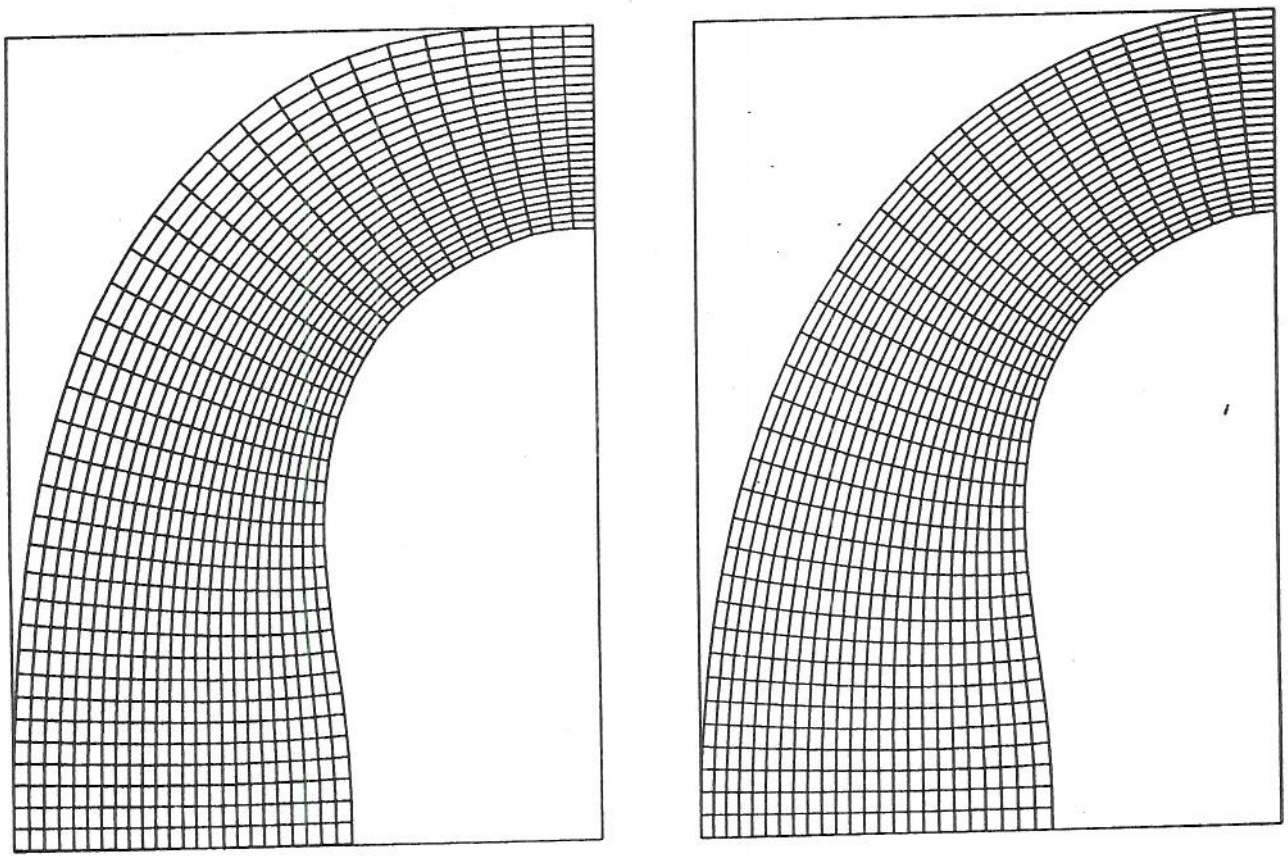


Fig. 14 – Mapping for computation of a channel shape given channel wall



a. – Computation of flow in a given channel

b. – Computation for an "optimum" channel shape

Fig. 15 – Channel flow solutions

Appendix 1 - Example codes

The code written to produce Fig. 4 is first presented. This is followed by a sample control file for the generalized version of SCGRID, and finally the subroutine in SCGRID that computes the integral of the Schwartz-Christoffel Transformation.

SCGRID itself is quite a long code, because it is generalized to many types of input. The basic structure of the code is the same as the code given below. It accepts the input from a file, creates an intermediate grid with the proper stretching, applies the transformation, then, if required, reflects the grid and computes the potential flow solution. There are two tricky additions, though. The first is the subroutine which computes the integral of the Schwartz-Christoffel Transformation, which is given on pages A.11 and A.12. The other is the computation of the values of the a_i; the algorithm for this is outlined on page 6.

SCGRID is run entirely by instructions from a file. Sample input files which fully illustrate how to use this code are available. The output can be accepted by two different VKI plotting programs : < gridplot.f > and < plot.f >. A sample input file is included below. Note that these files have been fully commented to shown exactly what modifications are allowed. It is hoped that the wide variety of example files (all the files necessary to produce the output in this report, and more) will illustrate how to use all of the options currently included in SCGRID. Further, the program SCGRID itself is unprotected and well documented. Users are encouraged to further adapt it to suit their own special needs.

The following program computes the conformal mapping shown in figure 2.

```
C ******************************************************************
C **                                                              **
C **  THIS 2-D GRID GENERATOR WAS WRITTEN BY THOMAS R. BEWLEY IN   **
C **  SPRING, 1990 AT THE von KARMAN INSTITUTE FOR FLUID DYNAMICS  **
C **                                                              **
C ******************************************************************
C
      PARAMETER  IMAX=100, JMAX=100,  MAXIT=30,
     &           PI=3.14159265358979, CONDMAX=30

      REAL*4     US(IMAX,JMAX),VS(IMAX,JMAX),GRTYPS

      REAL*8  X(IMAX,JMAX),Y(IMAX,JMAX),U(IMAX,JMAX),V(IMAX,JMAX),
     &        ICOEF(CONDMAX),VORIGIN,PORIGIN,RHORIGIN,
     &        RHO(IMAX,JMAX),RHOU(IMAX,JMAX),RHOV(IMAX,JMAX),
     &        RHOE(IMAX,JMAX),EPSILON

      INTEGER       I,J,K,ICOND,IDUMMY(40),VDIRECTION,POTENTIAL

      CHARACTER*80  INFILE,UVFILE,XYFILE,XYFILE1,POTFILE,
     &              TITLEUV(4),TEXT

C
C  THIS SECTION GETS THE NAME OF THE DATA FILE
C
      WRITE(6,100)
100   FORMAT(///,
     1 9X,'*******************************************************'//,
     2 9X,'                        C M G R I D 1'//,
     3 9X,'This program generates a grid for flow over a step via '/,
     4 9X,'a conformal mapping of a uniform flow onto the desired '/,
     5 9X,'domain.  The program takes data from a <*.cm1> file.'//,
     6 9X,'The program can also create a potential flow solution.'/,
     7 9X,'Written by Thomas Bewley at the von Karman Institute.'//,
     8 9X,'*******************************************************')
      WRITE (6,115)
115   FORMAT(///,'  Enter complete name of input file: ',$)
      READ(5,'(A30)') INFILE
      OPEN(UNIT=35,FILE=INFILE,STATUS='OLD')
C
C  THIS SECTION GETS THE DATA FROM THE FILE
C
      CALL READLINE(35,TEXT)
      READ(TEXT,*) H
      CALL READLINE(35,TEXT)
      READ(TEXT,*) JJ
      CALL READLINE(35,TEXT)
      READ(TEXT,*) U(1,1),V(1,1)
      READ(35,*)  (U(1,J),V(1,J),J=2,JJ)

      CALL READLINE(35,TEXT)
      READ(TEXT,*) ICOND
```

```
      CALL READLINE(35,TEXT)
      READ(TEXT,*) ICOEF(1)
      READ(35,*) (ICOEF(L),L=2,ICOND)
      CALL READLINE(35,TEXT)
      READ(TEXT,*) II
      CALL READLINE(35,TEXT)
      READ(TEXT,*) LL

      CALL READLINE(35,TEXT)
      READ(TEXT,'(A80)') XYFILE
      CALL READLINE(35,TEXT)
      READ(TEXT,'(A80)') XYFILE1

      CALL READLINE(35,TEXT)
      READ(TEXT,*) POTENTIAL
      IF (POTENTIAL.EQ.1) THEN
        CALL READLINE(35,TEXT)
        READ(TEXT,'(A80)') POTFILE
        CALL READLINE(35,TEXT)
        READ(TEXT,*) VDIRECTION
        CALL READLINE(35,TEXT)
        READ(TEXT,*) VORIGIN
        CALL READLINE(35,TEXT)
        READ(TEXT,*) PORIGIN
        CALL READLINE(35,TEXT)
        READ(TEXT,*) RHORIGIN
        ENDIF
      CLOSE(35)
C
C  COMPUTE THE POSITIONS Y(1,J) HERE
C
      DO 170 I=1,II
        A=(I-1.)/(II-1.)
        Y(I,1)=0.
        DO 160 L=1,ICOND
          B=1.
          DO 150 K=2,L
            B=B*A
150         CONTINUE
          Y(I,1)=Y(I,1)+LL*ICOEF(L)*B
160       CONTINUE
170     CONTINUE
C
C  COMPUTE THE POSITIONS X(1,J) HERE
C
      DO 180 J=1,JJ
        CALL RTFLSP(X(1,J),U(1,J),V(1,J),H,-5,5,.0001)
180   CONTINUE
C
C  THIS SECTION COMPUTES A RECTANGULAR GRID
C
      DO 190 J=1,JJ
        DO 185 I=1,II
          X(I,J)=X(1,J)
          Y(I,J)=Y(I,1)
185       CONTINUE
190     CONTINUE
C
```

```
C  THE UV GRID POINTS ARE NOW COMPUTED.
C
      DO 200 J=1,JJ
        DO 205 I=1,II
          CALL CM(U(I,J),V(I,J),H,X(I,J),Y(I,J))
205       CONTINUE
200     CONTINUE
C
C  THIS SECTION COMPUTES THE POTENTIAL FLOW SOLUTION
C
      IF (POTENTIAL.EQ.1) THEN
        WRITE(6,*)'Now computing the potential flow solution ...'
        EPSILON=.001
        DELTAX=0.
        DELTAY=0.
        WRITE(6,*)' VDIRECTION,EPSILON',VDIRECTION,EPSILON
        IF (VDIRECTION.EQ.1) DELTAY=EPSILON
        IF (VDIRECTION.EQ.2) DELTAY=-EPSILON
        IF (VDIRECTION.EQ.3) DELTAX=-EPSILON
        IF (VDIRECTION.EQ.4) DELTAX=EPSILON
        B2=VORIGIN*VORIGIN/2.+PORIGIN/RHORIGIN
        WRITE(6,*)' B2,DELTAX,DELTAY: ',B2,DELTAX,DELTAY
        I=2
669     J=JJ/2
        CALL INTEGRAL(X(I,J),Y(I,J),X(I,J)+DELTAX,Y(I,J)+DELTAY,
     &                      MR,MI,COR,AC,OMEGA,A1,B1,2)
        WRITE(6,*)' EPSILON ',EPSILON
        A2=VORIGIN*SQRT(A1*A1+B1*B1)/EPSILON
        WRITE(6,*)' A2: ',A2
        DO 680 I=1,II
         DO 685 J=1,JJ
          IF (VDIRECTION.EQ.2.AND.I.EQ.1) GOTO 685
          IF (VDIRECTION.EQ.4.AND.INTGRID.EQ.1.AND.J.EQ.1) GOTO 685
          IF (VDIRECTION.EQ.3.AND.INTGRID.EQ.1.AND.J.EQ.JJ) GOTO 685
          CALL INTEGRAL(X(I,J),Y(I,J),X(I,J)+DELTAX,Y(I,J)+DELTAY,
     &                        MR,MI,COR,AC,OMEGA,A1,B1,2)
          A=A2*EPSILON/SQRT(A1*A1+B1*B1)
          B=ATAN2(B1,A1)
          RHO(I,J)=RHORIGIN
          RHOU(I,J)=RHORIGIN*A*COS(B)
          RHOV(I,J)=RHORIGIN*A*SIN(B)
C         RHOE(I,J)=RHORIGIN*(2.5*B2-.75*A*A)
          RHOE(I,J)=10000
685       CONTINUE
680      CONTINUE
        END IF
C
C  THIS SECTION SAVES THE GRID
C
      OPEN(UNIT=1,FILE=UVFILE1,STATUS='NEW')
      WRITE(1,*) II,JJ
      WRITE(1,*) ((U(I,J),I=1,II),J=1,JJ),((V(I,J),I=1,II),J=1,JJ)
      CLOSE(UNIT=1)
C
C  THIS SECTION SAVES THE POTENTIAL FLOW SOLUTION FOR DXPLOT.F
C
      IF (POTENTIAL.EQ.1) THEN
        OPEN(UNIT=1,FILE=POTFILE,FORM='FORMATTED',STATUS='NEW')
```

```
801       FORMAT(1X,A)
          WRITE(1,802) II,JJ,1,II,0,0.
802       FORMAT(1X,5I5,F10.5)
          WRITE(1,803) ((SNGL(U(I,J)),SNGL(V(I,J)),I=1,II),J=1,JJ)
803       FORMAT(7(1X,E15.7))
          WRITE(1,803) ((SNGL(RHO(I,J)), SNGL(RHOU(I,J)),
     &            SNGL(RHOV(I,J)),SNGL(RHOE(I,J)),I=1,II),J=1,JJ)
          WRITE(1,804) 0.,0.,0.,1.4,1.,1.,1.,1.
804       FORMAT(1X,8F15.7)
          REWIND(1)
          CLOSE(1)
          END IF
        END
C
C  THIS SUBROUTINE USES A ROOT FALSE POSITION SCHEME TO COMPUTE THE
C  INVERSE TRANSFORMATION.
C
      SUBROUTINE RTFLSP(RTFLS,U,V,H,X1,X2,XACC)

      CALL CM(F1,F2,H,X1,0.)
      FL=U-V-F1+F2
      CALL CM(F1,F2,H,X2,0.)
      FH=U-V-F1+F2
      IF(FL*FH.GT.0.) PAUSE 'Root must be bracketed for RTFLSP.'
      IF(FL.LT.0.)THEN
        XL=X1
        XH=X2
      ELSE
        XL=X2
        XH=X1
        SWAP=FL
        FL=FH
        FH=SWAP
      ENDIF
      DX=XH-XL
      DO 11 J=1,MAXIT
        RTFLS=XL+DX*FL/(FL-FH)
        CALL CM(F1,F2,H,X1,0.)
        F=U-V-F1+F2
        IF(F.LT.0.) THEN
          DEL=XL-RTFLS
          XL=RTFLS
          FL=F
        ELSE
          DEL=XH-RTFLS
          XH=RTFLS
          FH=F
        ENDIF
        DX=XH-XL
        IF(ABS(DEL).LT.XACC.OR.F.EQ.0.) RETURN
11    CONTINUE
      PAUSE 'RTFLSP exceed maximum iterations'
      END
C
C  THIS SUBROUTINE EVALUATES THE INVERSE COSH OF Z
C
       SUBROUTINE INVCOSH(W,Z)
```

```
      COMPLEX*16  Z,W,TMP,TMP1

      TMP=Z*Z-1
      CALL MTH$CDSQRT(TMP1,TMP)
      TMP=TMP1+Z
      CALL MTH$CDLOG(W,TMP)
      END
C
C  THIS SUBROUTINE EVALUATES THE CONFORMAL MAPPING FUNCTION AT A
C  POINT Z (SUBJECT TO THE PARAMETER H) AND RETURNS THE VALUE IN W
C
      SUBROUTINE CM(WR,WI,H,ZR,ZI)

      REAL*8       ZR,ZI,H,WR,WI
      COMPLEX*16   Z,W,TMP,TMP1,DCMPLX
      REAL*8       MTH$DREAL,MTH$DIMAG

      Z=DCMPLX(ZR,ZI)
      TMP=Z*Z-1
      CALL MTH$CDSQRT(TMP1,TMP)
      CALL INVCOSH(TMP,Z)
      W=TMP+TMP1
      W=W*H/PI
      WR=MTH$DREAL(W)
      WI=MTH$DIMAG(W)
      END
C
C  THIS SUBROUTINE READS IN LINES FROM THE DATA FILE, IGNORING THOSE
C  LINES WHICH BEGIN WITH 'CC'
C
      SUBROUTINE READLINE(NTUNI,TEXT)

      CHARACTER*80 TEXT

2000  READ (NTUNI,'(A80)') TEXT
      IF (TEXT(1:2).EQ.'CC') GOTO 2000

      RETURN
      END
```

The following is a control file for the program SCGRID.

```
CC
CC  The conventional wedge, theta=40, alpha=10
CC
CC  This is an <*.scg> data file for the program <scgrid>.
CC  This file may be used as a format to run <scgrid> for arbitrary
CC  bodies with a finite number of straight segments.
CC
CC  Enter a point on the ray coming in from infinity to the body.
CC        X          Y
         -5.         0.
CC  Enter the number of corners corresponding to: a(i) < -1.
          0
CC  Enter the x and y coordinates of these corners enumerated TOWARDS
CC  the body.
CC
CC  Enter the number of corners on the body, corresponding to:
CC  -1. <= a(i) <= 1.
          2
CC  Enter the x and y coordinates of these corners enumerated in a
CC  CLOCKWISE sense on the body around the origin.  (The first is
CC  assigned a=-1 and the last is assigned a=1)
CC
CC        X          Y
 -.8390996          0.
          0.   .1763270
CC
CC  Enter the number of corners corresponding to: a(i) > 1.
          0
CC  Enter the x and y coordinates of these corners enumerated AWAY
CC  from the body.
CC        X          Y
CC
CC  Enter a point on the ray going out to infinity from this body
CC        X          Y
          0.         5.
CC
CC  Enter the type of the intermediate grid:
CC      1 - Confocal ellipses and hyperbola
CC      2 - Horizontal and Vertical lines
CC  INTGRID
        1
CC
CC  Enter the number of steps for the integral of the transformation
CC  between each point.
CC   STEPS
      50
CC  Enter the maximum number of iterations to find the a(i) and M
CC   ITERMAX
      15
```

```
CC  Note:  "polynomial stretching" used below is applied in the
CC   intermediate plane.  This has to be adjusted to give the desired
CC   stretching in the transformed plane.  All polynomials entered
CC   must equal 0. for index=0. and 1. for index=1.
CC ---------------------------------------------------------------
CC      ENTER THE STRETCHING FUNCTIONS FOR THE DESIRED GRID DENSITIES
CC ---------------------------------------------------------------
CC  Enter the number of coefficients for the polynomial streching
CC  function for the grid lines intersecting the x-axis in the
CC  intermediate plane between -1 and 1.
CC  Concentration of this stretching function near index=0. gives
CC  grid density near x<=1 and concentration near index=1. gives
CC  density near x>=-1.  Enter 0 for x-intercepts given by the
CC  inverse transform of the corners (entered above) corresponding to
CC  -1 < a(i) < 1.  Take JCOND = - (# of conditions) if this
CC  stretching fn is to be used between each a(i) in this region for
CC  a total of JJ pts.
CC   JCOND
       4
CC  Enter these coefficients, starting with the constant term.
CC  {Comment out this section if JCOND=0}
CC   JCOEF
       0.
       0.
       3.
      -2.
CC  Enter the resolution in this region
CC  {Comment out this section if JCOND=0}
CC     JJ
      100
CC --------------------
CC  Enter the number of coefficients for the polynomial streching
CC  function for the grid lines intersecting the x-axis in the
CC  intermediate plane to the left of -1.
CC  Concentration near index=0. gives density near x<=-1.
CC  Enter 0 for x-intercepts given by the inverse transform of the
CC  corners corresponding to a(i) < 1 or if INTGRID=1.
CC  Take JCONDL = - (# of conditions) if this stretching fn is to be
CC  used between each a(i) in this region for a total of JJL pts.
CC    JCONDL
        0
CC  Enter these coefficients, starting with the constant term.
CC  {Comment out this section if JCONDL=0}
CC    JCOEFL
CC
CC  Enter the resolution in this region
CC  {Comment out this section if JCONDL=0}
CC     JJL
CC
CC  Enter the extent of the intermediate grid in this direction, or
CC  the (positive) distance of the last point to the corner a=-1.
CC  {Comment out this section if JCONDL<1}
CC     LLL
CC
```

```
CC ---------------------
CC  Enter the number of coefficients for the polynomial streching
CC  function for the grid lines intersecting the x-axis in the
CC  intermediate plane to the right of +1.
CC  Concentration near index=1. gives density near x>=1.
CC  Enter 0 for x-intercepts given by the inverse transform of the
CC  corners corresponding to a(i) > 1 or if INTGRID=1.
CC  Take JCONDR = - (# of conditions) if this stretching fn is to be
CC  used between each a(i) in this region for a total of JJR pts.
CC    JCONDR
        0
CC  Enter these coefficients, starting with the constant term.
CC  {Comment out this section if JCONDR=0}
CC    JCOEFR
CC
CC  Enter the resolution in this region
CC  {Comment out this section if JCONDR=0}
CC     JJR
CC
CC  Enter the extent of the intermediate grid in this direction, or
CC  the distance of the last point to the corner a=1.
CC  {Comment out this section if JCONDR<1}
CC     LLR
CC
CC ---------------------
CC  Enter the number of coefficients for the polynomial stretching
CC  function for the grid lines intersecting the y-axis in the
CC  intermediate plane.  Concentration near index=0. gives density
CC  near y>=0.  Enter 1 for a hyperbolic tangent stretching function.
CC  If INTGRID=1, you may also: A) enter 0 for the y-intercepts given
CC  by the ellipses which intersect the x-axis at the points which
CC  are the inverse transform of the corners corresponding to
CC  a(i) > 1., or B) enter -1 for the y-intercepts given by the
CC  ellipses which intersect the x-axis at the points which are the
CC  inverse transform of the corners corresponding to a(i) < 1.
CC    ICOND
      3
CC  Enter these coefficients, starting with the constant term
CC  If ICOND=1, enter the number of boundary layer thicknesses (as a
CC  real number) total in the vertical direction.
CC  {Comment out this section if ICOND<1}
CC    ICOEF
        0.
       1.5
       -.5
CC  Enter the resolution in this region
CC  {Comment out this section if ICOND<1}
CC      II
       100
CC  Enter the extent of the intermediate grid in the vertical
CC  direction
CC  {Comment out this section if ICOND<1}
CC      LL
       1.7
```

```
CC ------------------------------------------------------------------
CC  Is the grid to be reflected across the vertical axis?
CC  (0=no, 1=yes, 2=twice)  (If 1 or 2, the grid points should end on
CC  the vertical axis at this point)
CC    VERTREF
        0
CC  Is the grid to be reflected across the horizontal axis?
CC  (0=no, 1=yes)  (If so, the grid points should end on the
CC  horizontal axis at this point.)
CC    HORREF
        0
CC  Enter the names of the final grid file and the intermediate file,
CC  first for the program <gridplot>, then for the program <PLOT>.
../grid/w.40.10.decgri
scout/eh.w.40.10.dec.gri
scout/uv.w.40.10.dat
scout/eh.w.40.10.dat
CC
CC  Is the incompressible potential flow solution to be found?
CC  (0=no, 1=yes)
CC   POTENTIAL
      0
CC  If so, then:
CC  Enter the name of the output file for the program <gridplot>.
CC   POTFILE
CC
CC  Enter the direction of the velocity vector at the origin in the
CC  intermediate plane.
CC    (1=Up, 2=Down, 3=Left, 4=Right)
CC   VDIRECTION
CC
CC  Enter the magnitude of the velocity at the point corresponding to
CC  the origin in the intermediate plane.
CC   VORIGIN
CC
CC  Enter the pressure of the point corresponding to the origin in
CC  the intermediate plane.
CC   PORIGIN
CC
CC  Enter the density.
CC   RHORIGIN
CC
CC --------------------------- End of File ------------------------
```

The following subroutine computes the integral of the Schwartz-Christoffel Transformation.

```
C
C  ** THIS SUBROUTINE WRITTEN BY THOMAS R. BEWLEY, FEB. 1990 **
C
C  THIS SUBROUTINE COMPUTES THE INTEGRAL OF THE SCHWARTZ CHRISTOFFEL
C  TRANSFORMATION ACCORDING TO THE GENERAL FORMULA GIVEN BY DAVIS IN
C  AIAA 79-1463 (EQN 17).  FOR CONVENENCE, IT USES THE SAME VARIABLES
C  AS USED BY DAVIS.  IN SHORT, THIS PROCEDURE TAKES TWO VALUES OF
C  ZETA [ ZETA1(R & I) AND ZETA2(R & I) ], SPLITS IT UP EVENLY INTO A
C  GIVEN NUMBER OF STEPS, AND SUMS THE INCREMENT GIVEN BY EQN 17 WITH
C  THE GIVEN VALUE OF M AND THE GIVEN NUMBER OF CORNERS AT A(I) WITH
C  ANGLES ALPHA(I). THE RESULTING VALUE OF DELTA-Z IS RETURNED IN
C  RESULT(R & I)
C
      SUBROUTINE INTEGRAL(ZETA1R,ZETA1I,ZETA2R,ZETA2I,MR,MI,CORNERS,
     &                    A,ALPHA,RESULTR,RESULTI,STEPS)

      IMPLICIT  LOGICAL (A-Z)

      PARAMETER  CORMAX1=400, PI=3.14159265358979
      INTEGER STEPS,CORNERS
      REAL    ZETA1R,ZETA1I,ZETA2R,ZETA2I,MR,MI,A(CORMAX1),
     &        ALPHA(CORMAX1),RESULTR,RESULTI
      INTEGER M,I
      REAL    TEMPMR(CORMAX1),TEMPMI(CORMAX1),TEMPMPR(CORMAX1),
     &        TEMPMPI(CORMAX1),DUMMYR,DUMMYI,R,THETA,DUMMY,
     &        PRODUCTR,PRODUCTI,DELTAR,DELTAI

      RESULTR=0.
      RESULTI=0.
      DELTAR=(ZETA2R-ZETA1R)/(STEPS-1.)
      DELTAI=(ZETA2I-ZETA1I)/(STEPS-1.)
     ,IF ((DELTAR*DELTAR+DELTAI*DELTAI).EQ.0.) GOTO 1030

      DO 1005 I=1,CORNERS
        DUMMYR=ZETA1R-A(I)
        DUMMYI=ZETA1I
        R=SQRT(DUMMYR*DUMMYR+DUMMYI*DUMMYI)
        IF (R.NE.0.) THEN
            THETA=ATAN2(DUMMYI,DUMMYR)
          R=EXP(LOG(R)*(ALPHA(I)/PI+1.))
          END IF
        THETA=THETA*(ALPHA(I)/PI+1.)
        TEMPMR(I)=R*COS(THETA)
        TEMPMI(I)=R*SIN(THETA)
1005    CONTINUE

      DO 1010 M=1,STEPS-1
        PRODUCTR=1.
        PRODUCTI=0.
        DO 1011 I=1,CORNERS
          DUMMYR=ZETA1R+DELTAR*M-A(I)
          DUMMYI=ZETA1I+DELTAI*M
```

```
          R=SQRT(DUMMYR*DUMMYR+DUMMYI*DUMMYI)
          IF (R.NE.0.) THEN
              THETA=ATAN2(DUMMYI,DUMMYR)
            R=EXP(LOG(R)*(ALPHA(I)/PI+1.))
            END IF
          THETA=THETA*(ALPHA(I)/PI+1.)
          TEMPMPR(I)=R*COS(THETA)
          TEMPMPI(I)=R*SIN(THETA)
1011      CONTINUE
        DO 1012 I=1,CORNERS
          DUMMYR=TEMPMPR(I)-TEMPMR(I)
          DUMMYI=TEMPMPI(I)-TEMPMI(I)
          DUMMY   =PRODUCTR*DUMMYR-PRODUCTI*DUMMYI
          PRODUCTI=PRODUCTI*DUMMYR+PRODUCTR*DUMMYI
          PRODUCTR=DUMMY
          TEMPMR(I)=TEMPMPR(I)
          TEMPMI(I)=TEMPMPI(I)
1012      CONTINUE
        RESULTR=RESULTR+PRODUCTR
        RESULTI=RESULTI+PRODUCTI
1010    CONTINUE

      DO 1014 I=1,CORNERS-1
        DUMMY=DELTAR*DELTAR+DELTAI*DELTAI
        DUMMYR=(RESULTR*DELTAR+RESULTI*DELTAI)/DUMMY
        RESULTI=(RESULTI*DELTAR-RESULTR*DELTAI)/DUMMY
        RESULTR=DUMMYR
1014    CONTINUE
      DUMMY  =RESULTR*MR-RESULTI*MI
      RESULTI=RESULTI*MR+RESULTR*MI
      RESULTR=DUMMY
      DO 1020 I=1,CORNERS
        RESULTR=RESULTR/(ALPHA(I)/PI+1.)
        RESULTI=RESULTI/(ALPHA(I)/PI+1.)
1020    CONTINUE
1030  RETURN
      END
```