Lawrence Berkeley Laboratory

UNIVERSITY OF CALIFORNIA

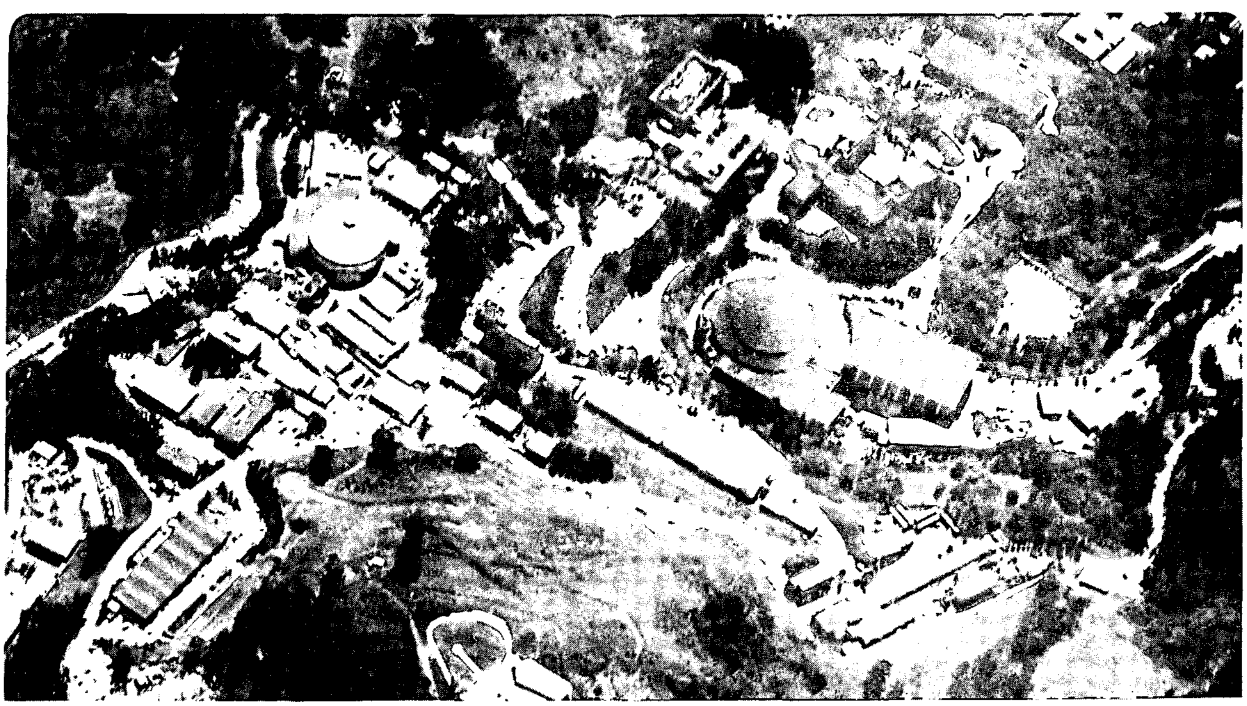
Physics Division

Mathematics Department

FLUID INTERFACES IN THE ABSENCE OF GRAVITY

M.C. Bainton
(M.A. Thesis)

May 1986



Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

LBL-21665

FLUID INTERFACES IN THE ABSENCE OF GRAVITY[1]

Mary Catherine Bainton

Department of Mathematics and Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

M.A. Thesis

May 1986

[1]Supported in part by the Applied Mathematical Sciences Subprogram of the Office of Energy Research, U.S. Department of Energy under contract DE-AC03-76SF00098.

Table of Contents

Acknowledgements

I am deeply indebted to Paul Concus and Ole Hald, without whom this thesis would not have been possible. Thanks are also due to Randolph Bank for his helpful telephone conversations on the use of PLTMG, and to Mark Rinker and Darren Miner for their invaluable advice and support.

This work was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy under contract DE-AC03-76SF00098.

Notation

a radius of smaller circle of curvilinear trapezoid

b radius of circle or radius of larger circle of curvilinear trapezoid

B Bond number (dimensionless capillary constant)

h straight side of curvilinear trapezoid bounding Ω^*

H Lagrange multiplier or mean curvature

$\vec{n}$ outward unit normal to Σ

R radius of critical arc

u solution to capillary surface problem

$\vec{W}$ vector field

β angle

γ contact angle with which fluid surface meets cylinder walls

Γ curve separating Ω into Ω^* and its complement

Ω cross-section of cylinder

Ω^* subset of Ω, bounded by Γ and Σ^*

ϕ angle of curvilinear trapezoid between axis of symmetry and a perpendicular from straight side

Φ functional depending on Γ and γ

Σ boundary of Ω

Σ^* boundary of $\Omega^* \cap \Sigma$

$|\cdot|$ length or area of $\cdot$

1. Introduction

Consider a fluid in a right cylinder of general cross-section. In the presence of the earth's gravitational field, a fixed volume of most liquids will form a stationary surface with the surrounding atmosphere [16,18]. In the absence of gravity, a stationary surface may or may not exist, depending on the cross-section of the cylinder and the contact angle between the surface and the cylinder walls [7].

Concus, Finn, and others (see [6,7,8,11,12,13,14,15] and the references therein) have considered this problem of existence. They have introduced a functional on curves embedded in the cross-section and shown that a solution exists if and only if this functional is strictly positive. Furthermore, Finn has put restrictions on the locally minimizing curves of this functional [13] and shown that a solution exits if and only if the local minima are all strictly positive [14]. These locally minimizing curves depend on the angle of contact between the surface and the cylinder walls. Concus and Finn [8,13] have studied the locally minimizing curves for cylinders of various cross-sections. In particular, Finn [13] has identified a unique critical curve, and accompanying critical contact angle, for the trapezoid. We conjecture that the curvilinear trapezoid has a similar unique critical curve.

The purpose of this paper is to study numerically the solution to the capillary surface equation in the absence of gravity for cylinders with curvilinear trapezoid cross-section, and also to determine computationally the critical contact angle at which a solution ceases to exist. We use the the PLTMG (Piecewise Linear Triangle Multigrid [4]) program to accomplish this. As test cases, we also study numerical solutions by PLTMG to this problem on circular and trapezoidal domains. We compare the numerical solution for the circle to the known exact solution, and that for the trapezoid to numerical solutions obtained by a program developed by Brown [5] and Roytburd [17].

Numerical results are given in section 4. Sections 2 and 3 contain general background material. In section 2, we present the general gravity-free capillary surface problem. In section 3, we discuss the subsidiary variational problem for the functional mentioned above.

2. The Capillary Surface Problem

2.1 Introduction

In this section, we derive the equations describing the height of a capillary surface in the absence of gravity. We then consider solutions for cylinders with circular, trapezoidal, and curvilinear cross-sections.

The Laplace-Young equation [16,18] for a stationary capillary surface between two fluids (generally a gas and a liquid) in a right cylinder of general cross-section Ω is

$$\nabla \cdot \frac{\nabla u}{\sqrt{1+|\nabla u|^2}} = Bu + 2H \quad \text{in } \Omega \tag{1}$$

$$\frac{\nabla u}{\sqrt{1+|\nabla u|^2}} \cdot \vec{n} = \cos\gamma \quad \text{on } \Sigma \tag{2}$$

where our notation is as follows [7]:

B Bond number (dimensionless capillary constant), depends on acceleration due to gravity, difference between gas and liquid densities, and gas-liquid surface tension;

H Lagrange multiplier, mean curvature when $B = 0$, depends on shape, volume, and γ;

$u(x, y)$ height of surface ;

γ contact angle with which fluid meets cylinder walls, depends on liquid, gas, and wall material;

Ω cross-section of cylinder;

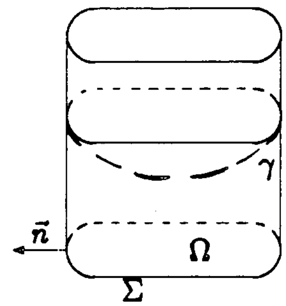


Figure 1: Cylinder of general cross-section Ω.

Σ boundary of Ω, continuous everywhere, differentiable except at finitely many corners;

$\vec{n}$ outward unit normal to Σ;

$|\cdot|$ area or length of $\cdot$.

Figure 1 illustrates $\vec{n}$, γ, Ω, and Σ.

In the absence of gravity, B is 0, and the problem becomes

$$\nabla \cdot \frac{\nabla u}{\sqrt{1+|\nabla u|^2}} = 2H \quad \text{in } \Omega$$

$$\frac{\nabla u}{\sqrt{1+|\nabla u|^2}} \cdot \vec{n} = \cos\gamma \quad \text{on } \Sigma,$$

in which case H is the (constant) mean curvature of the surface. The divergence theorem then gives us

$$\int_\Omega \nabla \cdot \frac{\nabla u}{\sqrt{1+|\nabla u|^2}} = \int_\Sigma \frac{\nabla u}{\sqrt{1+|\nabla u|^2}} \cdot \vec{n} = |\Sigma| \cos\gamma$$

$$\int_\Omega \nabla \cdot \frac{\nabla u}{\sqrt{1+|\nabla u|^2}} = \int_\Omega 2H = 2H|\Omega|,$$

which implies

$$H = \frac{|\Sigma| \cos\gamma}{|\Omega|}.$$

Hence, the capillary surface problem in the absence of gravity is

$$\nabla \cdot \frac{\nabla u}{\sqrt{1+|\nabla u|^2}} = \frac{|\Sigma| \cos\gamma}{|\Omega|} \quad \text{in } \Omega \tag{3}$$

$$\frac{\nabla u}{\sqrt{1+|\nabla u|^2}} \cdot \vec{n} = \cos\gamma \quad \text{on } \Sigma. \tag{4}$$

It can be shown that any stationary solution will be unique up to addition by a constant [7] and that the solution will be symmetric about any axis of symmetry in Ω [9]. Without loss of generality, we consider only $0 \leq \gamma < \pi/2$. The case of $\pi/2 < \gamma \leq \pi$ can be considered by looking at $-u$ and $-H$. The solution is identically constant for γ equal to $\pi/2$ [7,14].

2.2 Circle

A cylinder with circular cross-section is one of the few shapes for which a closed-form solution is known. If Σ is a circle of radius b, the solution to equations (3) and (4) is the portion of the lower hemisphere of radius $b/\cos\gamma$ given by [7]

$$u(x,y) = \text{constant} - \sqrt{\frac{b^2}{\cos^2\gamma} - (x^2+y^2)}, \qquad x^2+y^2 \leq b^2 \tag{5}$$

where, for example, the constant may be determined by the height of the solution at a particular point or by the prescribed volume of fluid.

2.3 Trapezoid

Equation (5) also gives the solution if Σ is a polygon circumscribing the circle

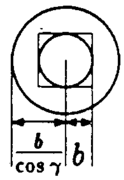


Figure 2: Square circumscribing circle of radius b.

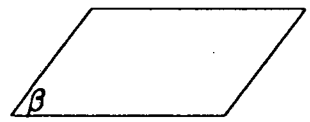


Figure 3: Parallelogram with smaller interior angle β.

of radius b such that no vertex lies outside the concentric circle of radius $b/\cos\gamma$ [7]. This configuration is shown for the square in figure 2.

A solution is known to exist for cylinders of parallelogramic cross-section when $\frac{\beta}{2} + \gamma \geq \frac{\pi}{2}$, where β is the smaller interior angle, as shown in figure 3. Finn [11] proved this by showing that a solution exists for a cylinder of general cross-section Ω if and only if a vector field $\vec{W}(\vec{x})$ exists in the closure of Ω such that

$$\nabla \cdot \vec{W} = \frac{|\Sigma|}{|\Omega|} \quad \text{in } \Omega$$

$$\vec{n} \cdot \vec{W} = 1 \quad \text{on } \Sigma$$

$$|\vec{W}| < \frac{1}{\cos\gamma} \quad \text{in } \Omega.$$

Then Finn [11,13] constructed such a vector field for the parallelogram with $\frac{\beta}{2} + \gamma \geq \frac{\pi}{2}$. Hence, a solution exists for a rectangle whenever γ is greater than or equal to $\pi/4$. However, Finn [13] has shown that the same result does not hold

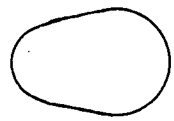


Figure 4: Curvilinear trapezoid.

for the general trapezoid. He constructed trapezoids which are arbitrarily close to a given rectangle, but for which no solution exists for an arbitrary contact angle. Roytburd [17] studied a specific example of this phenomenon numerically.

2.4 Curvilinear Trapezoid

A curvilinear trapezoid is a trapezoid with the parallel sides replaced by circular arcs joined differentiably onto the non-parallel sides, as shown in figure 4. This shape is of particular interest because a solution is known to exist for cylinders of curvilinear rectangular cross-section, but, given an arbitrary contact angle, a curvilinear trapezoid can be found for which no solution exists [10]. We consider existence criteria and numerical solutions for cylinders of curvilinear trapezoidal cross-section in sections 3 and 4, respectively.

3. The Subsidiary Variational Problem

3.1 Introduction

In this section, we introduce a functional on curves embedded in the cross-section of a cylinder. This functional is useful because a solution to the capillary surface problem in the absence of gravity exists if and only if all the local minima of this functional are strictly positive. We then consider existence criteria based on this functional for cylinders of circular, trapezoidal, and curvilinear trapezoidal cross-section.

Consider the functional [6]

$$\Phi(\Gamma) \equiv |\Gamma| - (|\Sigma^*| - \frac{|\Sigma||\Omega^*|}{|\Omega|}) \cos\gamma$$

where our notation is as follows:

γ contact angle with which fluid meets cylinder walls;

Γ curve separating Ω into Ω^* and its complement,

should be continuous everywhere, differentiable except at finitely many points (can actually be a system of rectifiable curves which, along with Σ^*, form the boundary of a finite number of connected, not necessarily disjoint Ω^* [14]);

Ω cross-section of cylinder;

Ω^* subset of Ω, bounded by Γ and Σ^*;

Σ boundary of Ω;

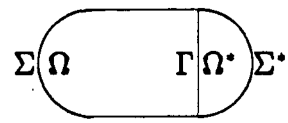


Figure 5: Ω divided by Γ.

Σ^* boundary of Ω^*, should be continuous everywhere,

differentiable except at finitely many points;

$|\cdot|$ area or length of $\cdot$.

Figure 5 illustrates Γ, Ω, Ω^*, Σ, and Σ^*.

Concus and Finn [6] have proved that if a stationary capillary surface exits in the absence of gravity in Ω, then Φ is strictly positive for all Γ. Giusti [15] has in effect proved that if $\Phi(\Gamma)$ is strictly positive for all Γ, then a solution exists. Hence, showing the existence of a solution is equivalent to showing that Φ is strictly positive for all Γ.

Finn [13,14] has shown that any curve Γ which locally minimizes Φ must consist of countably many arcs of circles of radius $R = \frac{|\Omega|}{|\Sigma|\cos\gamma}$, each of which should satisfy the following conditions:

be strictly smaller than a semi-circle,

curve into Ω^*,

not intersect other curves, except perhaps at a corner,

meet Σ with angle γ at both ends, as measured from inside Ω^*,

as shown in figure 6. Finn [14] has also proved that a solution exists if Φ does not achieve an absolute minimum. If a solution exists, then the infimum of Φ is

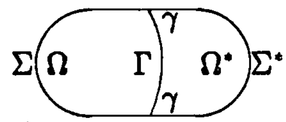


Figure 6: Γ dividing Ω and meeting Σ^* with angle γ.

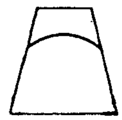


Figure 7: Critical Γ for the trapezoid.

zero [9,14]. (The existence of a solution implies that Φ is strictly positive. For any Ω, Φ will tend to zero on any sequence of $\{\Gamma_i\}$ which tend to a point on the boundary.) Hence, showing the existence of a solution is reduced to showing only that Φ is strictly positive for all of these locally minimizing Γ.

3.2 Circle

If Σ is a circle of radius b, a solution exists (see section 2.1). Hence, $\Phi(\Gamma)$ is greater than zero for all Γ.

3.3 Trapezoid

Finn [13] has considered all possible locally minimizing curves Γ for the cylinder of trapezoidal cross-section. He has shown that the arc which meets the nonparallel sides (with angle γ) and curves toward the top, as shown in figure 7, uniquely gives the lowest possible local minimum of Φ.

Roytburd [17] studied a trapezoid for which this lowest possible local minimum of Φ changes sign with γ. For the trapezoid with height 25, base length 2, and top length 1.3, he found that

$$sign(\Phi(\Gamma)) = sign(\gamma - \sim 57.6^o).$$

Hence, no solution exists for γ less than or equal to approximately 57.6^o.

3.4 Curvilinear Trapezoid

We conjecture that a result similar to that for the trapezoid holds for the curvilinear trapezoid. In other words, we conjecture that the arc which meets the straight sides (with angle γ) and curves toward the end of smaller radius, as shown in figure 8, gives the lowest possible local minimum of Φ.

The boundary length and area of a curvilinear trapezoid with smaller radius a, larger radius b, and angle ϕ between the axis of symmetry and a perpendicular from a straight side, as shown in figure 9, are

$$|\Sigma| = 2[b(\pi + \tan\phi - \phi) - a(\tan\phi - \phi)]$$

$$|\Omega| = b^2(\pi + \tan\phi - \phi) - a^2(\tan\phi - \phi).$$

To verify our conjecture about the location of the critical curve, we need to compare all possible local minima of Φ to that obtained from our conjectured curve.

Figure 8: Conjectured critical arc for the curvilinear trapezoid. (Case 0)

Case 0. For our conjectured critical curve we obtain the following values:

$$|\Gamma| = 2R(\phi - \gamma), \quad R = \frac{|\Omega|}{|\Sigma|\cos\gamma}$$

$$|\Sigma^*| = 2[R(\tan\phi\cos\gamma - \sin\gamma) - a(\tan\phi - \phi)]$$

$$|\Omega^*| = R^2(\tan\phi\cos^2\gamma - \sin\gamma\cos\gamma) - a^2(\tan\phi - \phi) - R^2(\phi - \gamma)$$

$$\Phi = R\{\phi - \gamma + \cos^2\gamma[(\frac{2a|\Sigma|}{|\Omega|} - \frac{a^2|\Sigma|^2}{|\Omega|^2})(\tan\phi - \phi) - \tan\phi] + \frac{\sin(2\gamma)}{2}\}.$$

It is geometrically clear that γ must be between 0 and ϕ. The value of Φ at γ equal to 0 is

$$\Phi|_{\gamma=0} = \frac{|\Omega|}{|\Sigma|}(\tan\phi - \phi)(\frac{2a|\Sigma|}{|\Omega|} - \frac{a^2|\Sigma|^2}{|\Omega|^2} - 1)$$

which is always negative or zero. The value of Φ at the limiting point γ equal to ϕ is

$$\Phi|_{\gamma=\phi} = \frac{|\Omega|}{|\Sigma|}\cos^2\phi(\tan\phi - \phi)\frac{a|\Sigma|}{|\Omega|}(2 - \frac{a|\Sigma|}{|\Omega|})$$

which is always positive or zero. (We have

$$\frac{a|\Sigma|}{|\Omega|} \leq \frac{2[b^2(\pi + \tan\phi - \phi) - ab(\tan\phi - \phi)]}{b^2(\pi + \tan\phi - \phi) - a^2(\tan\phi - \phi)} \leq 2$$

since a is less than b.) Thus Φ will always have at least one zero for γ between 0 and ϕ. Let h be the length of that part of the straight side lying between the

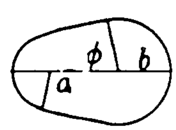


Figure 9: Curvilinear trapezoid with smaller radius a, larger radius b, and angle ϕ.

beginning of the smaller circle and Γ given by

$$h = \tan\phi(\frac{|\Omega|}{|\Sigma|} - a) - \frac{|\Omega|}{|\Sigma|}\tan\gamma.$$

Since h is a length, we must restrict ourselves to γ giving non-negative values of h. h is a monotonically decreasing function of γ since its first derivative with respect to γ is

$$\frac{dh}{d\gamma} = \frac{-|\Omega|}{|\Sigma|\cos^2\gamma},$$

which is negative for all γ between 0 and ϕ. Hence h will be zero at

$$\gamma_{max} = \arctan(\tan\phi(1 - \frac{a|\Sigma|}{|\Omega|}))$$

provided a is less than or equal to $|\Omega|/|\Sigma|$. We conjecture, on the basis of numerical experiments, that the zero of Φ between $\gamma = 0$ and $\gamma = \gamma_{max}$, if one exists, is unique.

We study a specific curvilinear trapezoid for which this conjectured lowest local minimum of Φ changes sign with γ. For the curvilinear trapezoid with axis of symmetry length 6.159, smaller radius .5, and larger radius 1 we find that the critical γ is approximately 30^o, as shown in figure 10. For this case, figure 10 indicates that the zero of Φ between $\gamma = 0$ and $\gamma = \gamma_{max}$ is unique.

Three locally minimizing arcs which can be ruled out are the following:

Case 1. If Γ consists of two arcs, each of 2β radians lying in Ω and with endpoints on the axis of symmetry, as shown in figure 11, then we obtain the following values:

$$|\Gamma| = 4\beta R$$

$$|\Sigma^*| = |\Sigma|$$

$$|\Omega^*| = |\Omega| - 2\beta R^2 + R^2 \sin(2\beta)$$

$$\Phi(\Gamma) = R(2\beta + \sin(2\beta)).$$

Hence, Φ is positive for all β between zero and $\frac{\pi}{2}$. Consequently, this curve need not be considered.

Case 2. If Γ meets Σ only on one straight side, as shown in figure 12, γ would have to be greater than $\pi/2$. We need not consider such γ (see section 2.1), and hence need not consider such Γ.

Case 3. If Γ consists of two arcs, each on separate sides of the axis of symmetry, and each meeting Σ at a straight side and at the smaller circle, curving toward the narrower end, as shown in figure 13, then the two angles of incidence can not be equal [9]. Consequently, we can discard this Γ as not being locally minimizing.

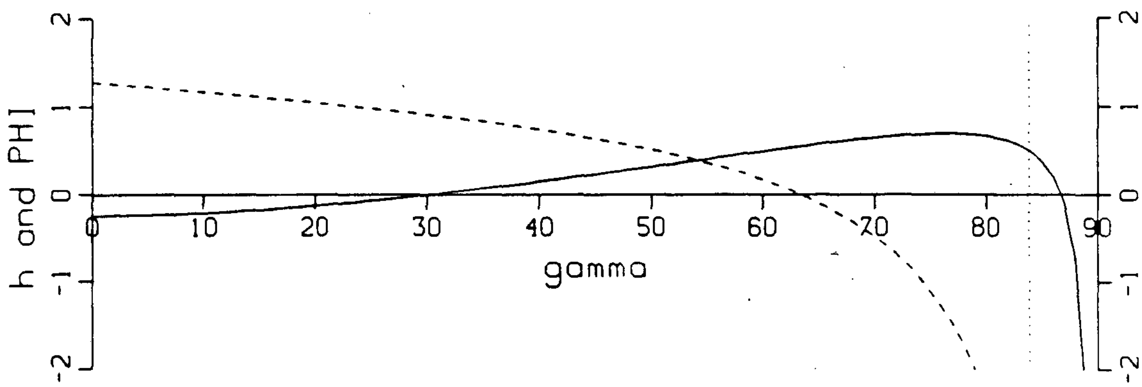


Figure 10: h (dashed) and Φ (solid) vs. γ (degrees) for curvilinear trapezoid with axis of symmetry length 6.159, smaller radius .5, and larger radius 1 ($\phi = 83.8°$).

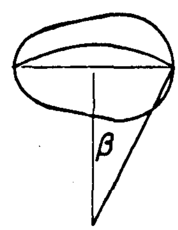


Figure 11: Γ meeting Σ at axis of symmetry. (Case 1)

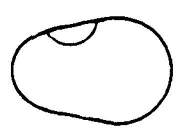


Figure 12: Γ meeting Σ at one straight side. (Case 2)

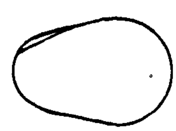


Figure 13: Γ meeting Σ at straight side and narrower end, curving toward narrower end. (Case 3)

4. Numerical Results from PLTMG

4.1 Introduction

In this section, we study numerically the solution to the capillary surface equation in the absence of gravity for cylinders with curvilinear trapezoidal cross-section, and also determine computationally the critical contact angle at which a solution ceases to exist. We use the PLTMG (Piecewise Linear Triangle Multi Grid) program to accomplish this. As test cases, we also study numerical solutions by PLTMG to this problem for circular and trapezoidal cross-sections.

The PLTMG program uses a continuous piecewise linear triangular finite element discretization and a multi-level iterative procedure [1,2,3,4] to solve nonlinear boundary value problems of the form

$$\nabla \cdot \vec{a}(x, y, u, \nabla u, \lambda) = f(x, y, u, \nabla u, \lambda) \text{ in } \Omega$$

$$u = g_1(x, y, \lambda) \text{ on } \Sigma_1 \subset \Sigma$$

$$\vec{a} \cdot \vec{n} = g_2(x, y, u, \lambda) \text{ on } \Sigma_2 = \Sigma - \Sigma_1$$

where the notation is the following [4] :

Ω is a connected region in the x-y plane,

$\vec{n}$ is the outward unit normal to Σ,

$\vec{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$,

a_1, a_2, f, g_1, g_2 are scalar functions,

λ is a scalar continuation parameter,

Σ is the boundary of Ω.

The program was designed for flexibility rather than speed. Starting with an initial triangulation supplied by the user or generated by the program, PLTMG can solve the problem with an optional continuation procedure on the coarsest grid, and adaptive and/or user-specified grid refinement. At each level, it can calculate error estimates of the accuracy in the H^1, L^2, and L^∞ norms. The adaptive procedure is based on H^1 error estimates [4]. Also, it can evaluate the solution and its gradient at user-specified points, study convergence if the exact solution is known, evaluate integrals involving the solution, and draw triangulations and solution surface and contour plots.

We used PLTMG with adaptive refinement and no continuation, after having revised it to allow the 'tacking down' of the first vertex. This was necessary because the solution to the capillary surface problem is unique only up to addition by a constant (see section 2.1). Specifying the value of the solution at a point, specifies the value of this additive constant. (Changes made to the program and subroutines and functions particular to our problem can be found in the appendix.) The H^1 error analyses in the following tables were all calculated by PLTMG. The times listed in the tables are the total execution times for the major functions in PLTMG and were returned by PLTMG in its output. These times are only approximate; they varied depending on which PLTMG options were used and the number of other time-sharers on the machine. These variations were sometimes by as much as a factor of two. The execution time entered

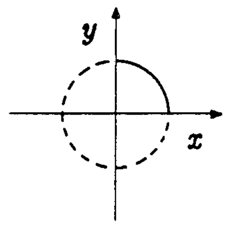


Figure 14: Orientation of the axes for the circle.

for each case in the tables is the smallest of the times used by PLTMG in that particular case for our runs.

4.2 Circle

To check the accuracy of the program, we considered the problem

$$\nabla \cdot \frac{\nabla u}{\sqrt{1+|\nabla u|^2}} = \frac{|\Sigma|\cos\gamma}{|\Omega|} \quad \text{in first quadrant} \cap \text{unit disk}$$

$$u(0,0) = 0$$

$$\frac{\nabla u}{\sqrt{1+|\nabla u|^2}} \cdot \vec{n} = \begin{cases} \cos\gamma & \text{on first quadrant} \cap \text{unit circle} \\ 0 & \text{on y-axis} \\ 0 & \text{on x-axis} \end{cases}$$

for γ equal to 0^o, 25^o, 50^o, and 75^o, where the axes are oriented as shown in figure 14. The exact solution is the lower hemispherical surface of equation (5).

The computed solutions compared well with the exact solutions after five levels, as can be seen from their plots along the positive x-axis in figures 16 and 17. PLTMG was particularly slow to approach a limiting solution for the limiting case of γ equal to 0^o, as can be seen from the error analysis in table 1. The estimated digits of accuracy in the H^1 norm were close to the actual digits, except for the case γ equal to 0^o. We tried the case γ equal to 0^o with different starting triangulations, from different initial guesses, and on different machines.

The numerical solutions for the early levels depended significantly on the initial discretization. The starting triangulations of figures 20 and 21 both have four triangles and six vertices. When they were both started with initial guess identically equal to zero, the triangulation of figure 20 approached a limiting solution more rapidly for the second and later levels, as can be seen from the plot of the value of the solution at the boundary point (1,0) against level in figure 15. The error analyses and times for these are given in tables 2 and 3.

As would be expected, the initial guess had little effect on the approach of the computed solution to the exact solution. (Any guess for which there is convergence of the program's Newton iteration should give the same solution for the same grid refinement.) When we started the triangulation of figure 20 with initial guess equal to the exact solution, the solutions were similar to those for initial guess identically equal to zero. All of the program runs described above were done on a VAX 11/780. As would be expected, execution times were shortened and solutions were similar on a VAX 8600.

γ	vertices	triangles	H^1 digits estimated	H^1 digits actual	least squares fit	execution time(sec)
0	2063	5460	.76	.33	.38	3042
25	1899	5248	1.7	1.7	.019	2577
50	2043	5452	1.9	1.9	.013	1510
75	2053	5444	1.9	1.9	.012	1173

Table 1: PLTMG error analysis and time for $\gamma = 0^o$, $25^o, 50^o, 75^o$ after 5 levels.

level	vertices	triangles	H^1 digits estimated	H^1 digits actual	least squares fit	execution time(sec)
1	6	4	.52	.09	-	1
2	29	56	.60	.11	.37	5
3	118	276	.67	.17	.40	47
4	510	1300	.75	.24	.40	277
5	2063	5460	.76	.33	.38	3042
6	8244	22144	.80	.47	.38	37657

Table 2: PLTMG error analysis and time for $\gamma = 0^o$, levels 1 through 6, figure 20 triangulation.

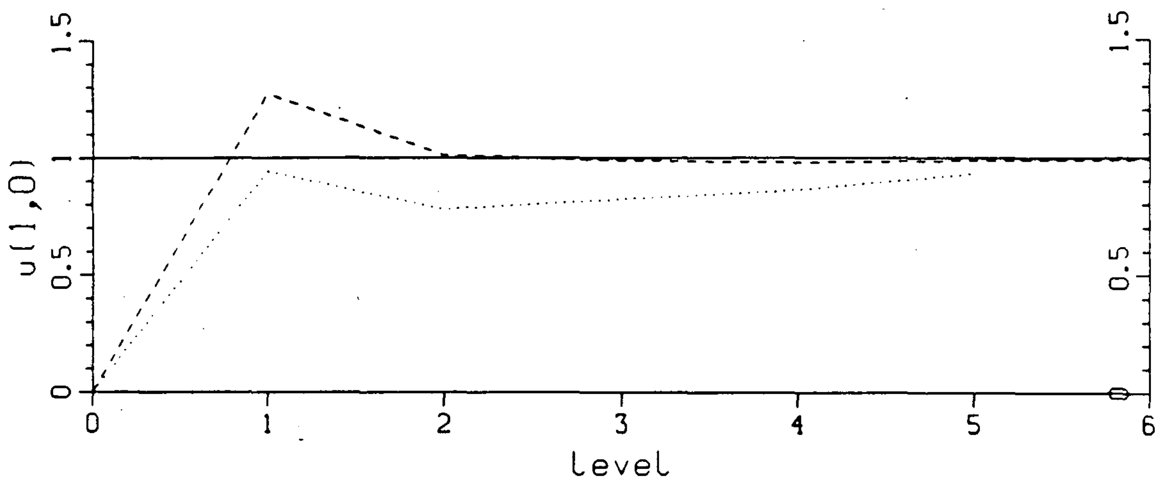


Figure 15: $u(1,0)$ vs. level for triangulation of figure 20 (dashed), triangulation of figure 21 (dotted), and exact solution (solid) for $\gamma = 0^o$

			H^1 digits		least	execution
level	vertices	triangles	estimated	actual	squares fit	time(sec)
1	6	4	.37	.09	-	1
2	27	60	.53	.12	.56	5
3	97	252	.64	.17	.50	27
4	417	1108	.63	.25	.50	303
5	1705	4588	.63	.36	.47	2110

Table 3: PLTMG error analysis and time for $\gamma = 0^o$, figure 21 triangulation.

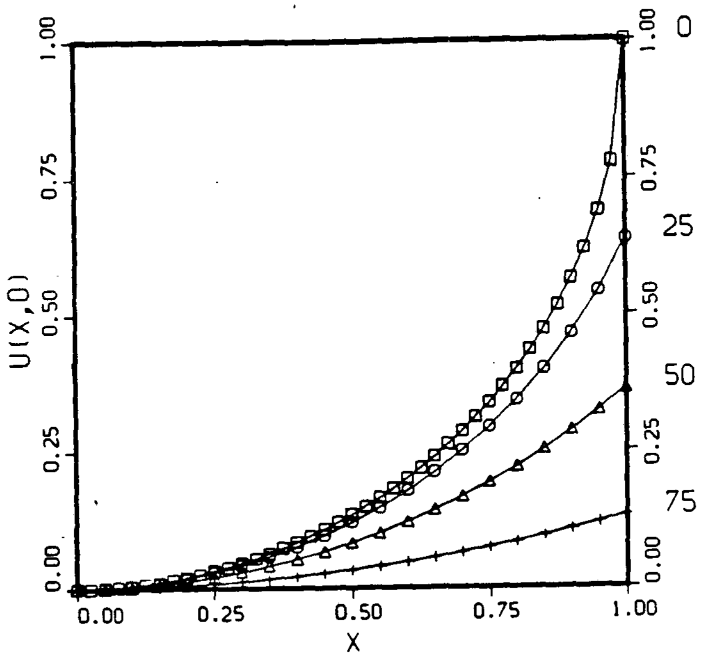


Figure 16: Exact $u(x,0)$ vs. x for $\gamma = 0^o, 25^o, 50^o, 75^o$.

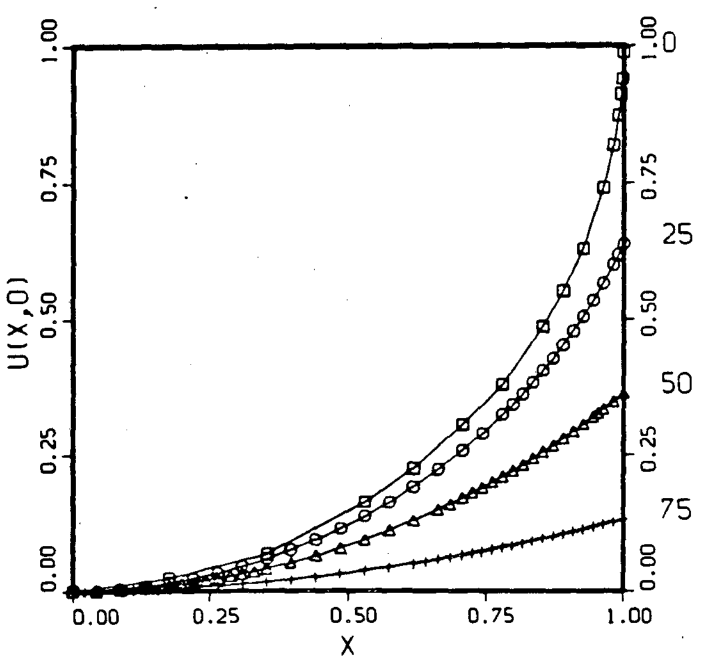


Figure 17: Computed $u(x,0)$ vs. x for $\gamma = 0^o, 25^o, 50^o, 75^o$ after 5 levels.

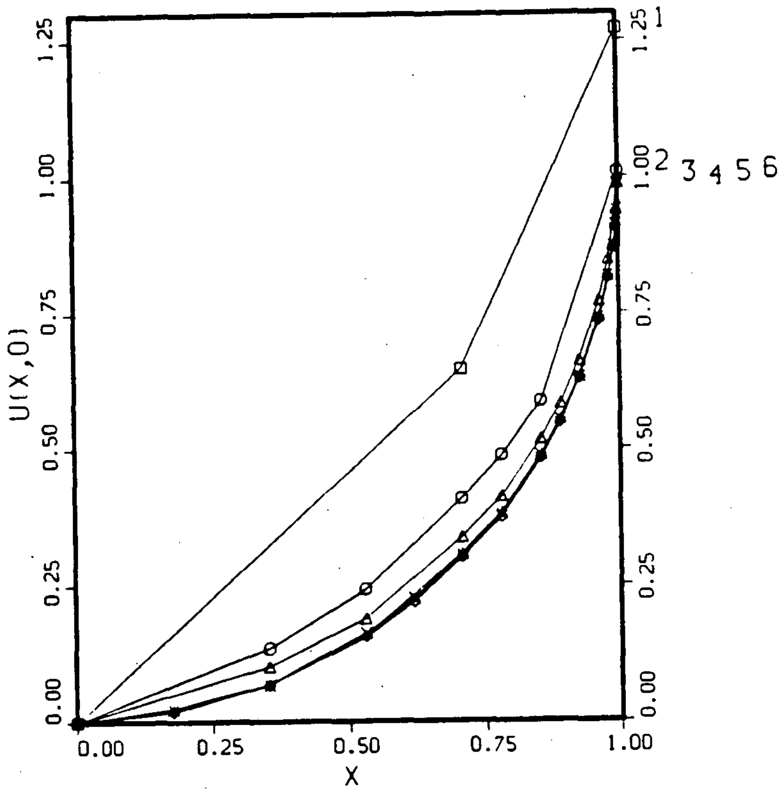


Figure 18: Computed $u(x,0)$ vs. x for $\gamma = 0^o$, levels 1 through 6, figure 20 triangulation.

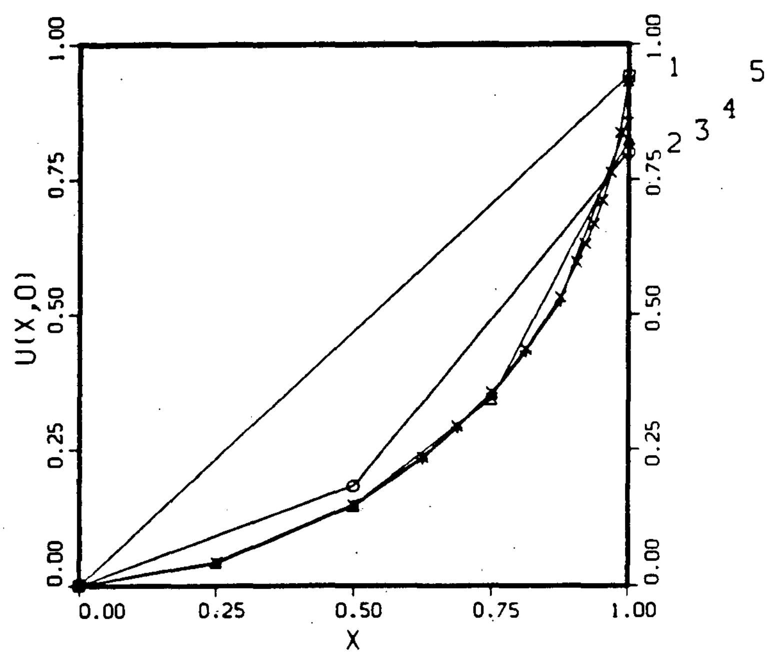


Figure 19: Computed $u(x,0)$ vs. x for $\gamma = 0^o$, levels 1 through 5, figure 21 triangulation.

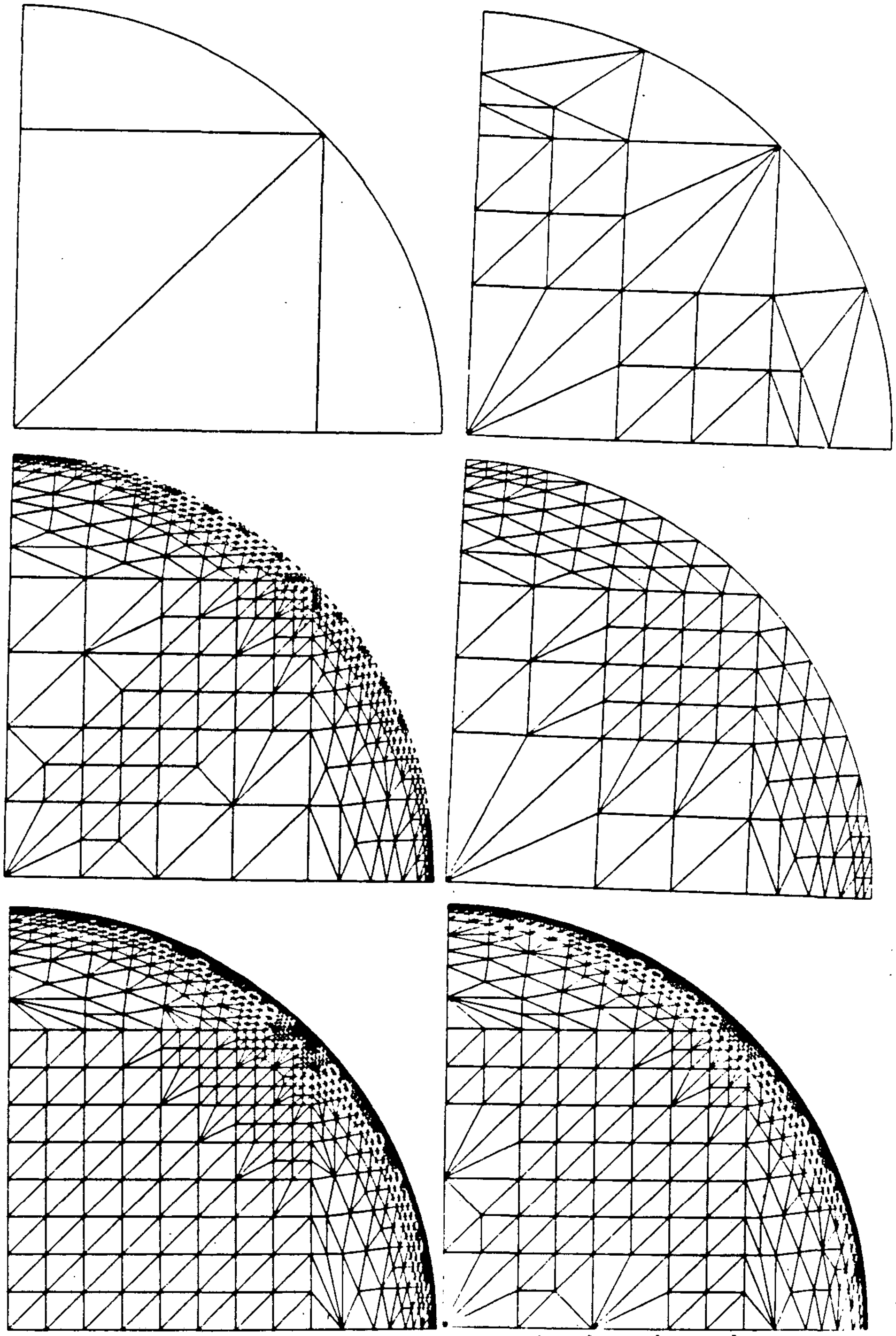


Figure 20: Triangulation for $\gamma = 0^o$, levels 1 through 6.

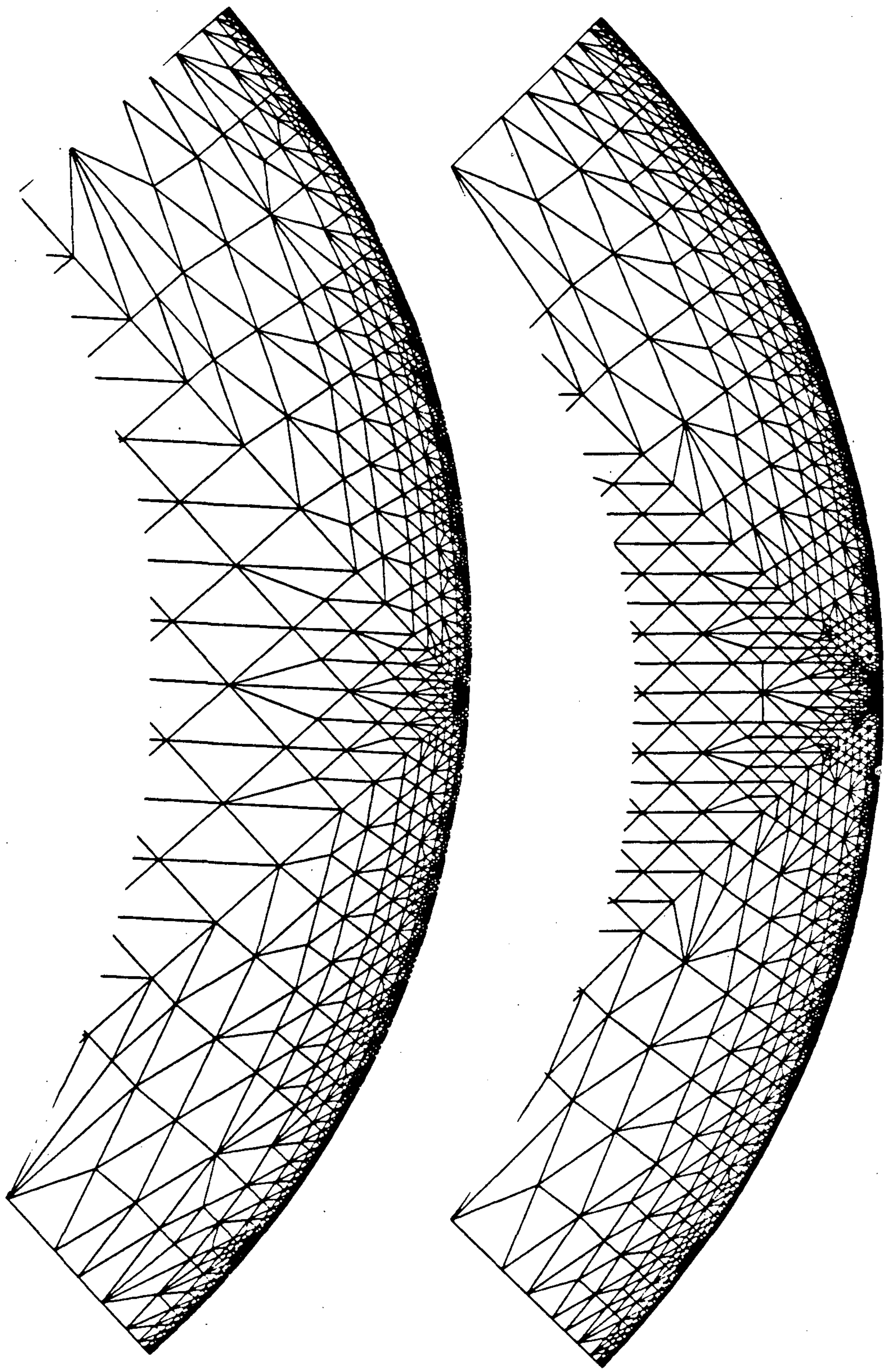


Figure 20: (con't) Triangulation for $\gamma = 0^o$, levels 5 and 6 magnified 2.4 times.

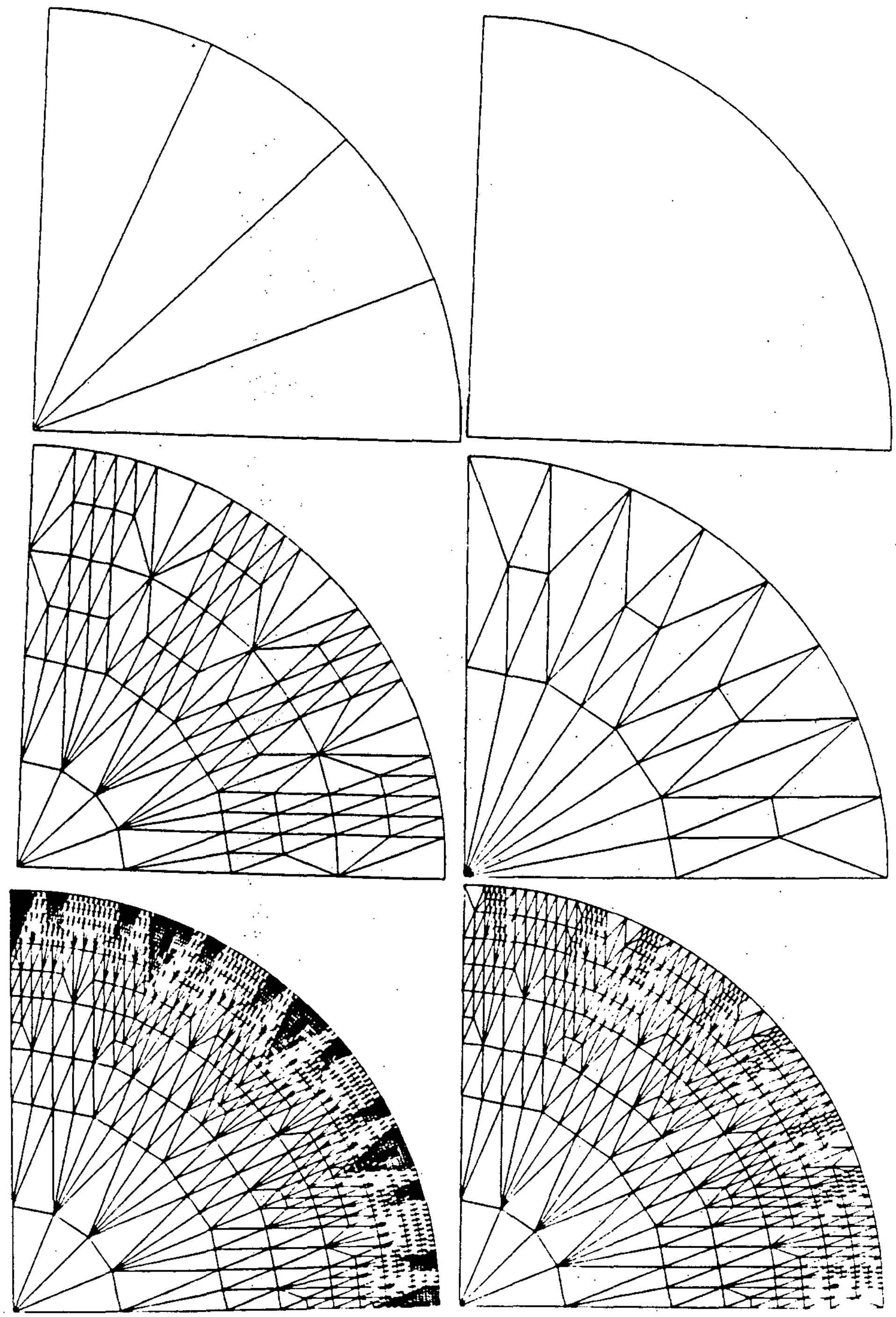


Figure 21: Alternate triangulation for $\gamma = 0°$, levels 0 through 5.

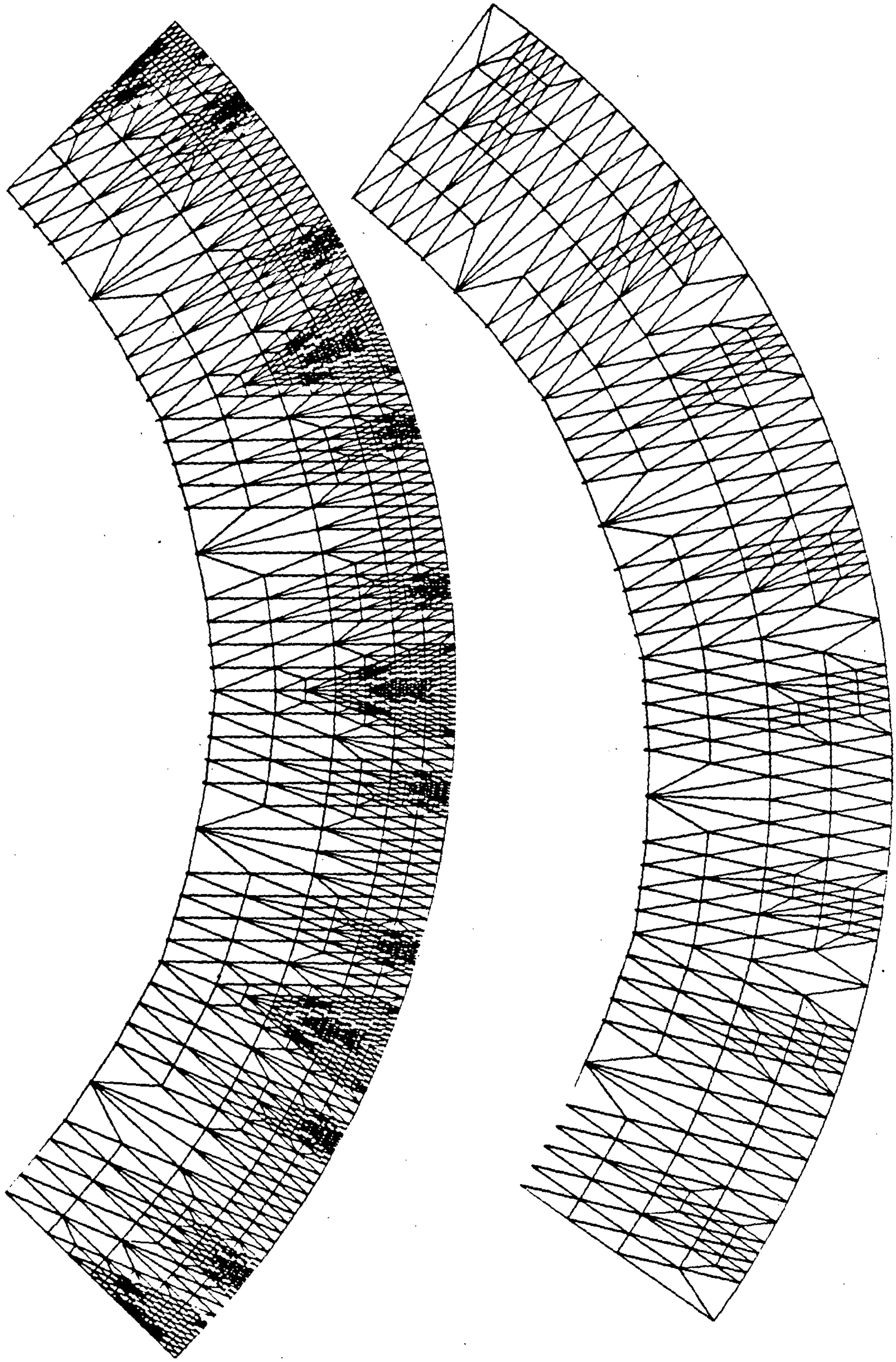


Figure 21: (con't) Triangulation for $\gamma = 0^o$, levels 4 and 5 magnified 2.4 times.

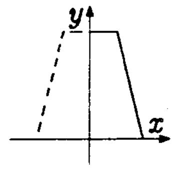


Figure 22: Orientation of the axes for the trapezoid.

4.3 Trapezoid

To check the accuracy of the program further, we considered the problem

$$\nabla \cdot \frac{\nabla u}{\sqrt{1+|\nabla u|^2}} = \frac{|\Sigma| \cos\gamma}{|\Omega|} \quad \text{in right half of trapezoid}$$

$$u(0,0) = 0$$

$$\frac{\nabla u}{\sqrt{1+|\nabla u|^2}} \cdot \vec{n} = \begin{cases} \cos\gamma & \text{on right half of trapezoid} \\ 0 & \text{on y-axis} \end{cases}$$

for γ equal to 58^o, height 25, base length 2, and top lengths 1.3, 1.4, 1.5, and 2., where the axes are oriented as shown in figure 22. The critical γ for top length equal to 1.3 is approximately 57.6^o (see section 3.3), so we approach a critical configuration as the top length decreases.

We compared the solutions from PLTMG to those from a program developed by Brown [5] and Roytburd [17] and modified by Jim Shearer and Jing Li. This program uses a biquadratic quadrilateral finite element method on a single fixed mesh. The solution along the y-axis computed by this program on a 5 x 50 mesh is shown in figure 24. Shorter run times and/or greater accuracy might be expected for this program because it uses a fixed mesh, was specifically designed for the capillary surface problem on trapezoidal domains, and uses biquadratic, rather than linear, elements.

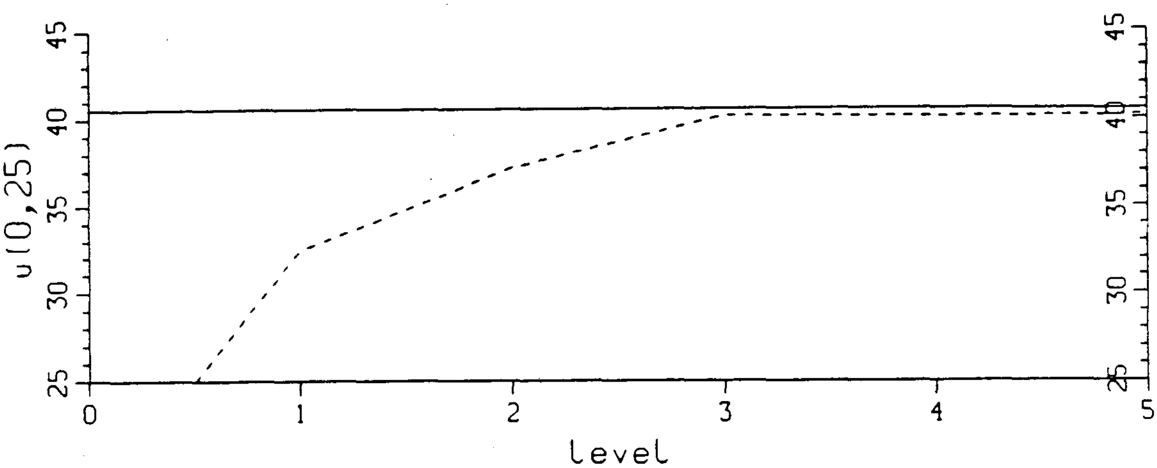


Figure 23: $u(0,25)$ vs. level for PLTMG (dashed) and Brown/Roytburd (solid), for top length 1.3.

The PLTMG solutions agreed with the fixed mesh solutions, although considerably longer run times were required. The solution along the y-axis computed by PLTMG is shown in figure 25. The error analysis and time are given in table 4.

The nearly critical case with top length equal to 1.3 required particularly long run times. The fixed mesh of Brown and Roytburd used approximately 48 seconds of cpu time to calculate the solution for the top length of 1.3. PLTMG used approximately 600 seconds of execution time to approach a limiting solution slightly less than that of the fixed mesh program on the third level. The value of the solution at the boundary point (0,25) is plotted versus level for top length equal to 1.3 in figure 23. There was little change in the computed value at this boundary point after level 3. The solutions computed by PLTMG were found to be sensitive to the starting triangulation. The solutions presented here for top length equal to 1.3 started from a triangulation clustered around the critical Γ, as shown in figure 27. We do this because ∇u is expected to be large along the critical Γ. The computed solution along the y-axis for levels 1 through 5 is shown in figure 26 for this case. The error analysis and time are given in table 5.

top length	vertices	triangles	estimated H^1 digits	execution time(sec)
2	9469	25636	2.4	1407
1.5	8405	22626	2.6	1403
1.4	8378	22522	2.7	1873
1.3	9377	26172	2.7	6853

Table 4: PLTMG error analysis and time for top lengths 2, 1.5, 1.4, and 1.3 after 5 levels.

level	vertices	triangles	estimated H^1 digits	execution time(sec)
1	32	30	1.5	4
2	142	286	1.7	19
3	568	1396	2.1	184
4	2283	6130	2.3	1081
5	9377	26172	2.7	6853

Table 5: PLTMG error analysis and time for top length 1.3.

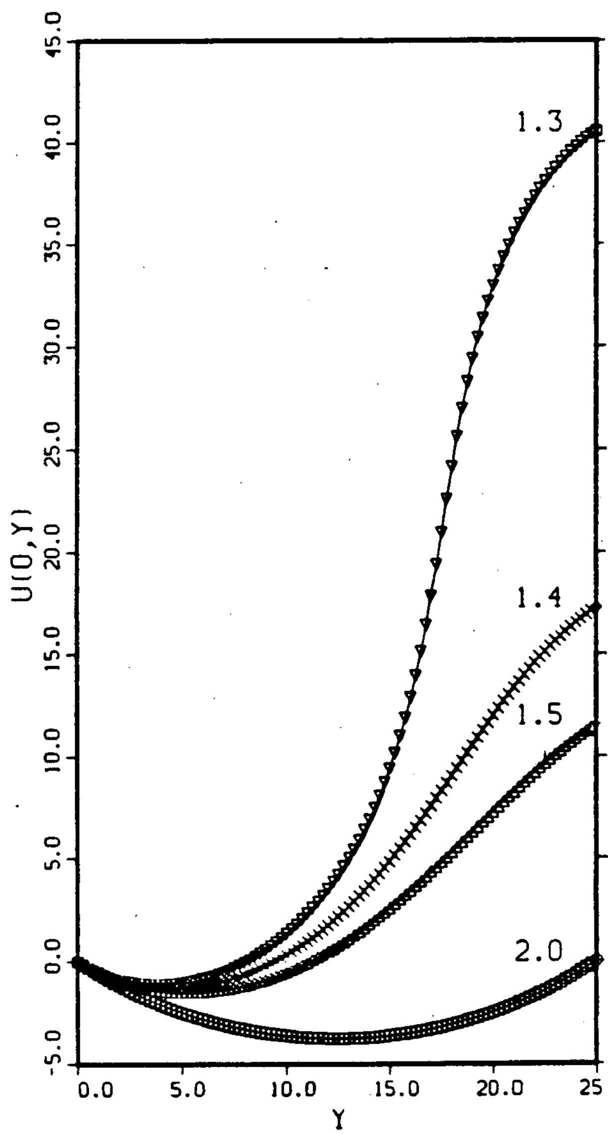


Figure 24: Brown/Roytburd $u(0, y)$ vs. y for top lengths 2, 1.5, 1.4, and 1.3., on a 5x50 mesh.

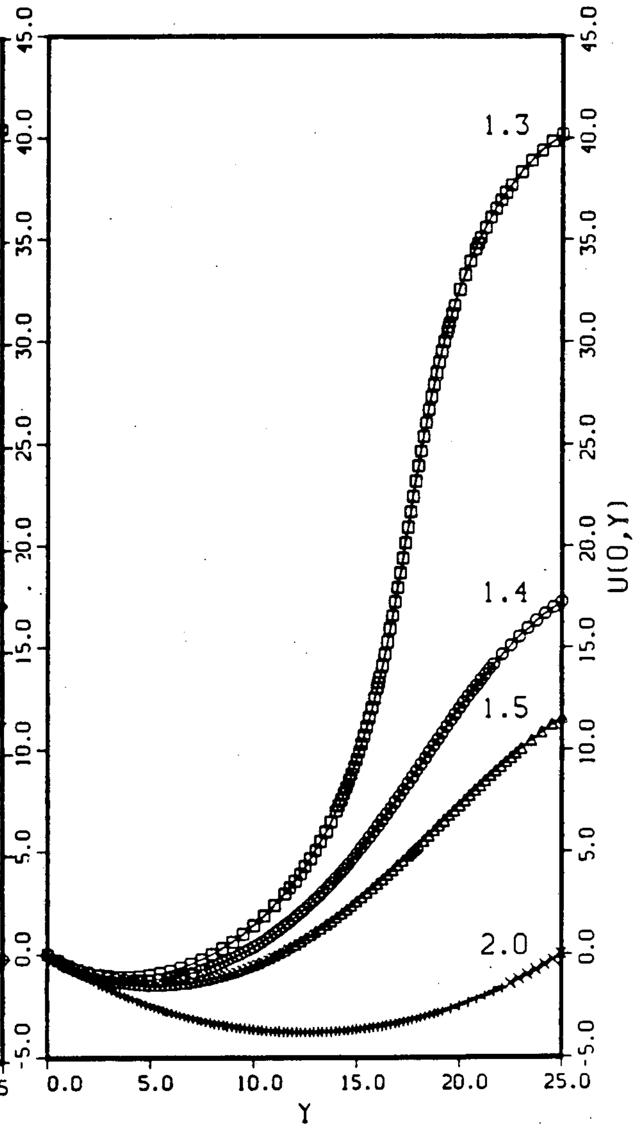


Figure 25: PLTMG $u(0, y)$ vs. y for top lengths 2, 1.5, 1.4, and 1.3 after 5 levels.

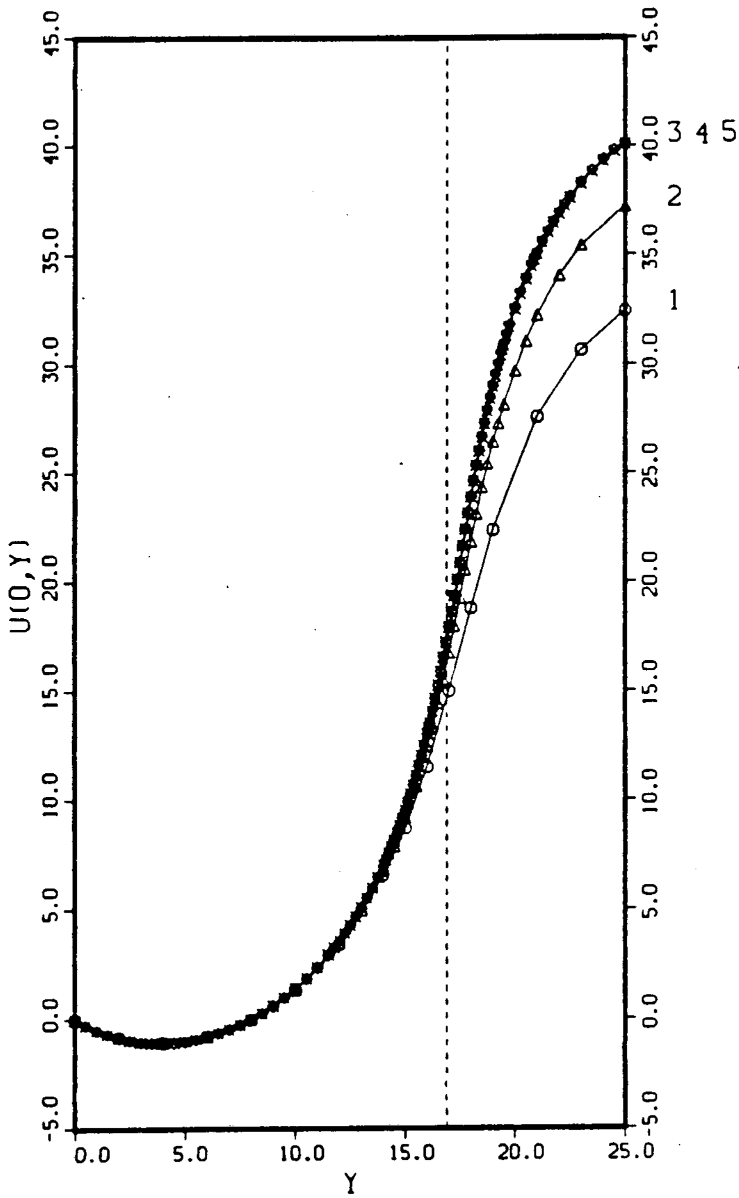


Figure 1: PLTMG $u(0,y)$ vs. y for top length 1.3, levels 0 through 5. Dashed line indicates critical $\Gamma \cap y$–axis.

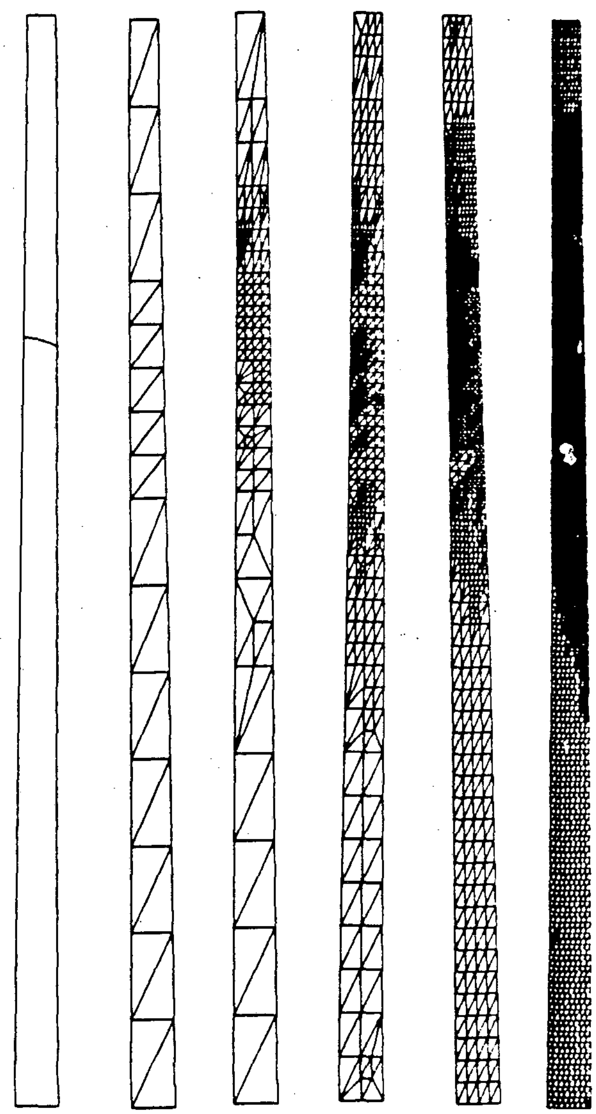


Figure 27: Triangulation for top length 1.3, levels 0 through 5. Critical Γ is drawn for level 0.

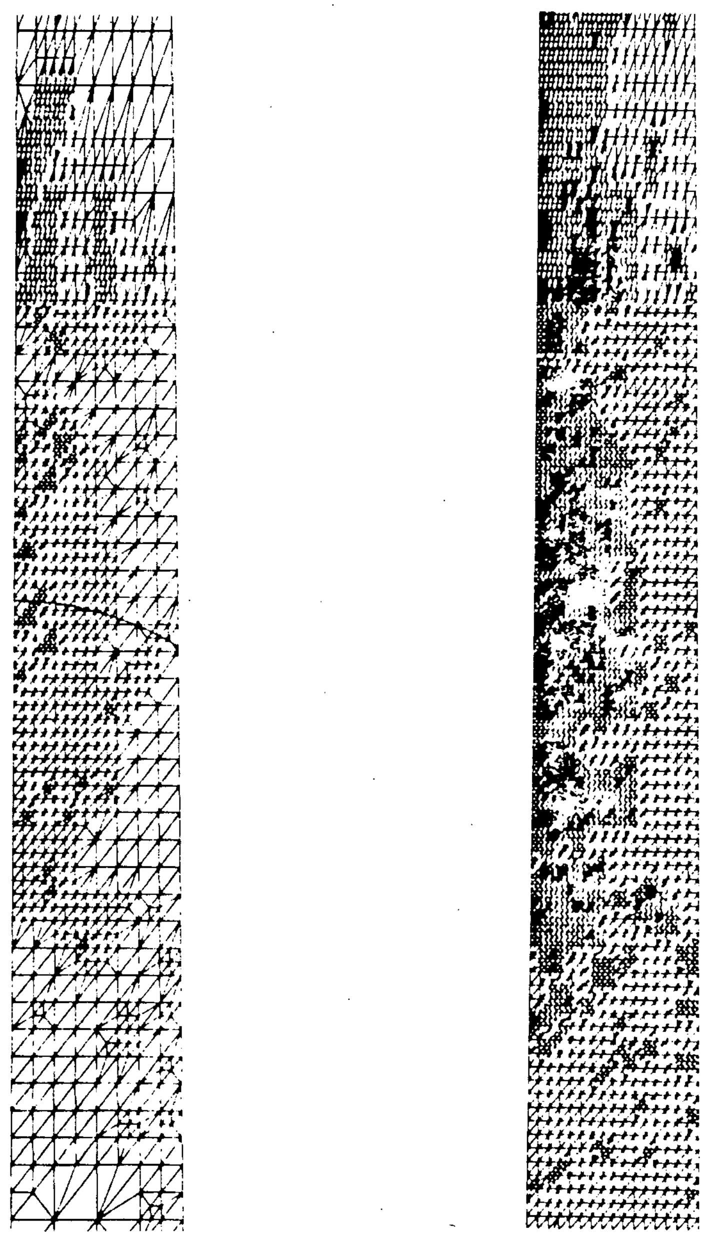


Figure 27: (con't) Triangulation for top length 1.3, levels 4 and 5 magnified 5 times about the critical Γ.

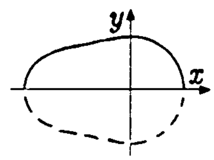


Figure 28: Orientation of the axes for the curvilinear trapezoid.

4.4 Curvilinear Trapezoid

In this final section, we address ourselves to the purpose of this paper—to numerically obtain the solution to the capillary surface equation in the absence of gravity for cylinders with curvilinear trapezoidal cross-section, and also to determine the critical contact angle at which a solution ceases to exist. We considered the problem

$$\nabla \cdot \frac{\nabla u}{\sqrt{1+|\nabla u|^2}} = \frac{|\Sigma|\cos\gamma}{|\Omega|} \quad \text{in top half of curvilinear trapezoid}$$

$$u(0,0) = 0$$

$$\frac{\nabla u}{\sqrt{1+|\nabla u|^2}} \cdot \vec{n} = \begin{cases} \cos\gamma & \text{on top half of curvilinear trapezoid} \\ 0 & \text{on x-axis} \end{cases}$$

for γ equal to $30.6^o, 40^o, 50^o$, and 80^o, axis of symmetry length 6.159, larger radius 1, and smaller radius .5, where the axes are oriented as shown in figure 28. The analytically derived critical γ for this configuration is approximately 30^o (see section 3.4), below which no solution exists. PLTMG was found to not converge for γ less than 30.6^o.

Again, PLTMG required lengthy run times to approach a limiting solution. The solution along the x-axis computed by PLTMG is shown in figure 30. The

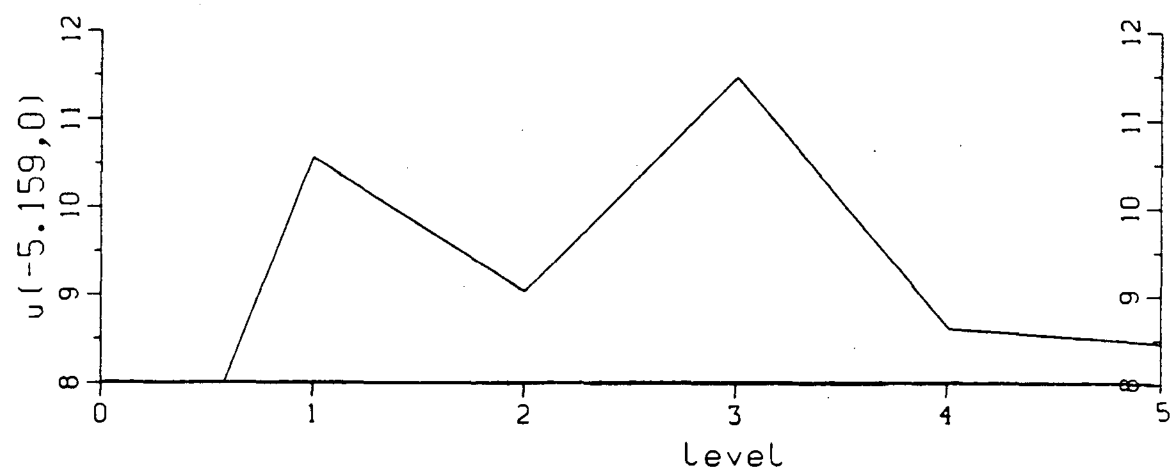


Figure 29: $u(-5.159, 0)$ vs. level for $\gamma = 30.6^o$.

error analysis and time are given in table 6. As was the case for the trapezoid, the maximum height of the solution increases as γ approaches the critical γ.

The nearly critical case with γ equal to 30.6^o required particularly long run times. We used a starting triangulation clustered around the critical Γ, as shown in figure 32. (∇u is expected to be large along the critical Γ.) The value of the solution at the leftmost boundary point is plotted versus level for γ equal to 30.6^o in figure 29. The solution value appears to settle at approximately 8.5, but oscillates before the fourth level. The computed solution along the x-axis for levels 1 through 5 is shown in figure 31 for this case. The error analysis and time are given in table 7. Surface and contour plots drawn by PLTMG for the fourth level are shown in figure 33.

The case of γ equal to 40^o consistently required unusually long run times as compared to the other cases, even that of the critical γ.

γ	vertices	triangles	estimated H^1 digits	execution time(sec)
80	3263	8747	1.9	1237
50	3602	9583	1.9	1575
40	4101	11047	1.9	4167
30.6	4202	11979	1.7	2118

Table 6: PLTMG error analysis and time for $\gamma = 30.6^o, 40^o, 50^o$, and 80^o after 5 levels.

level	vertices	triangles	estimated H^1 digits	execution time(sec)
1	15	13	.54	2
2	61	121	.46	16
3	237	603	.85	101
4	1017	2821	1.4	998
5	4202	11979	1.7	2118

Table 7: PLTMG error analysis and time for $\gamma = 30.6^o$.

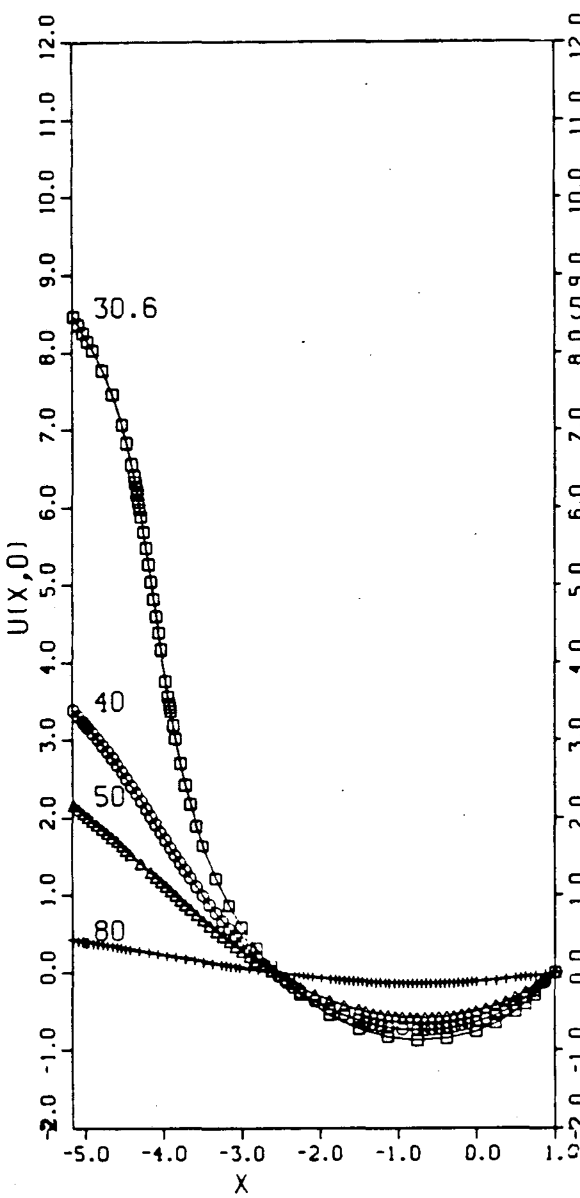


Figure 2: Computed $u(x,0)$ vs. x for $\gamma = 30.6^o, 40^o, 50^o$, and 80^o after 5 levels.

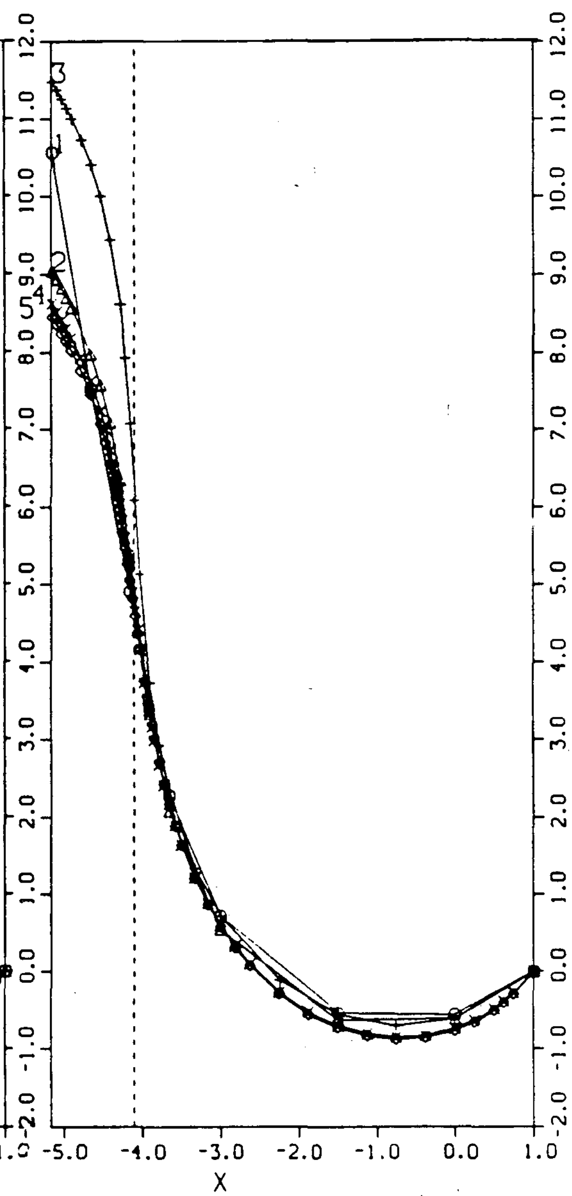


Figure 3: Computed $u(x,0)$ vs. x for $\gamma = 30.6^o$, levels 1 through 5. Dashed line indicates critical $\Gamma \cap x-$axis.

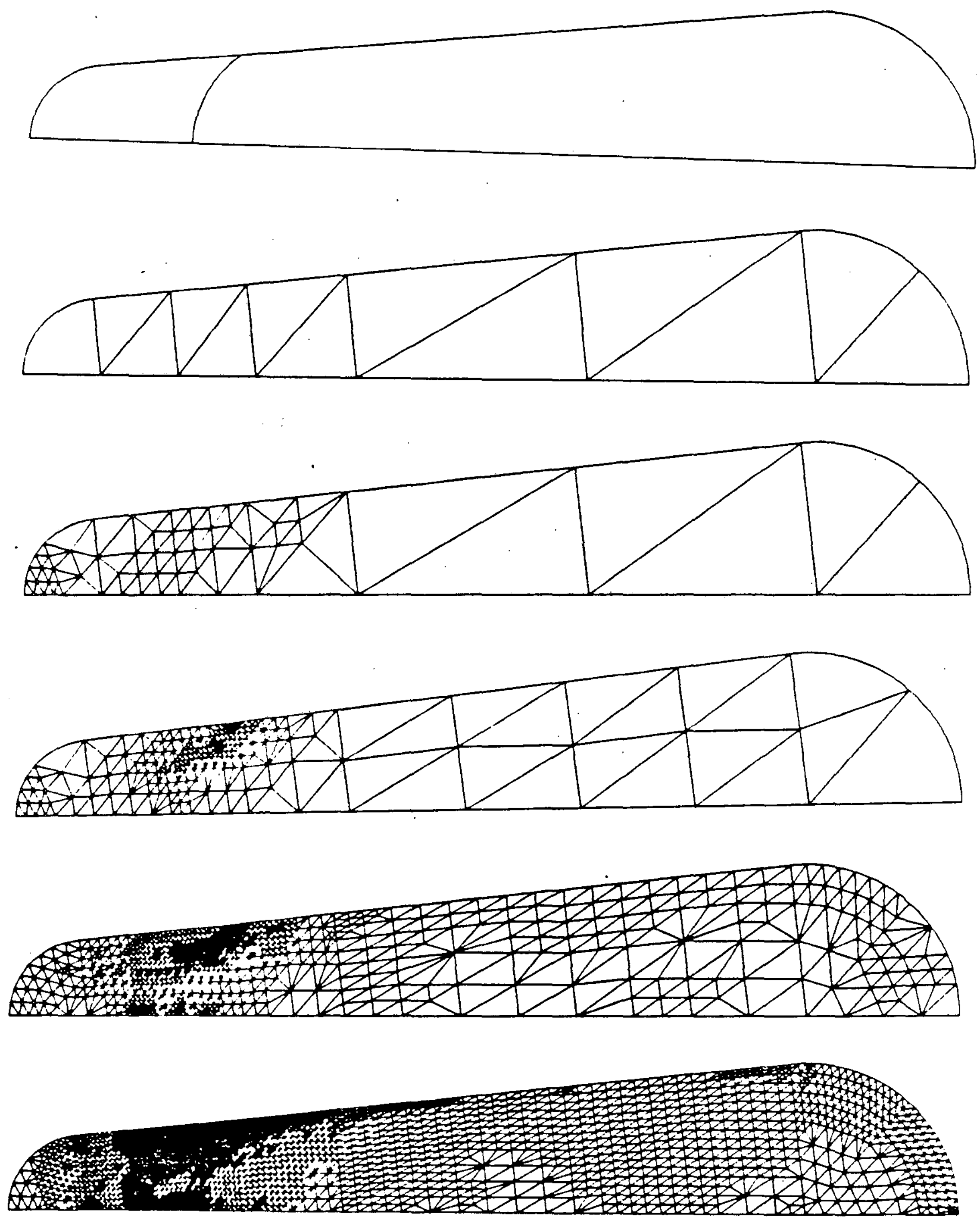
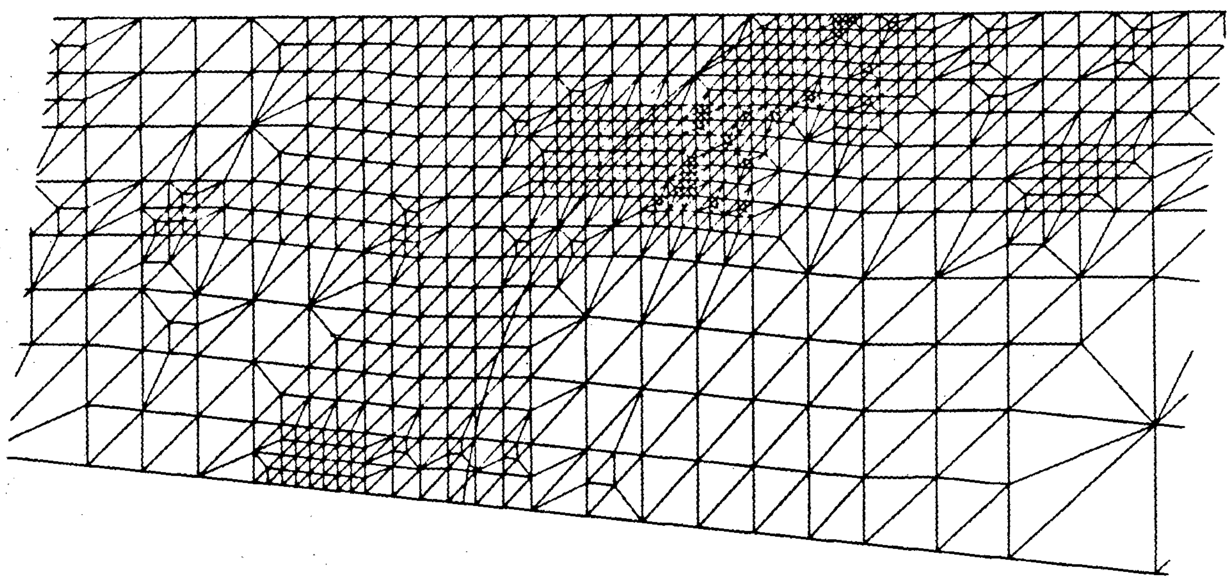


Figure 32: Triangulation for $\gamma = 30.6^{\circ}$, levels 0 through 5. Critical curve is drawn for level 0.



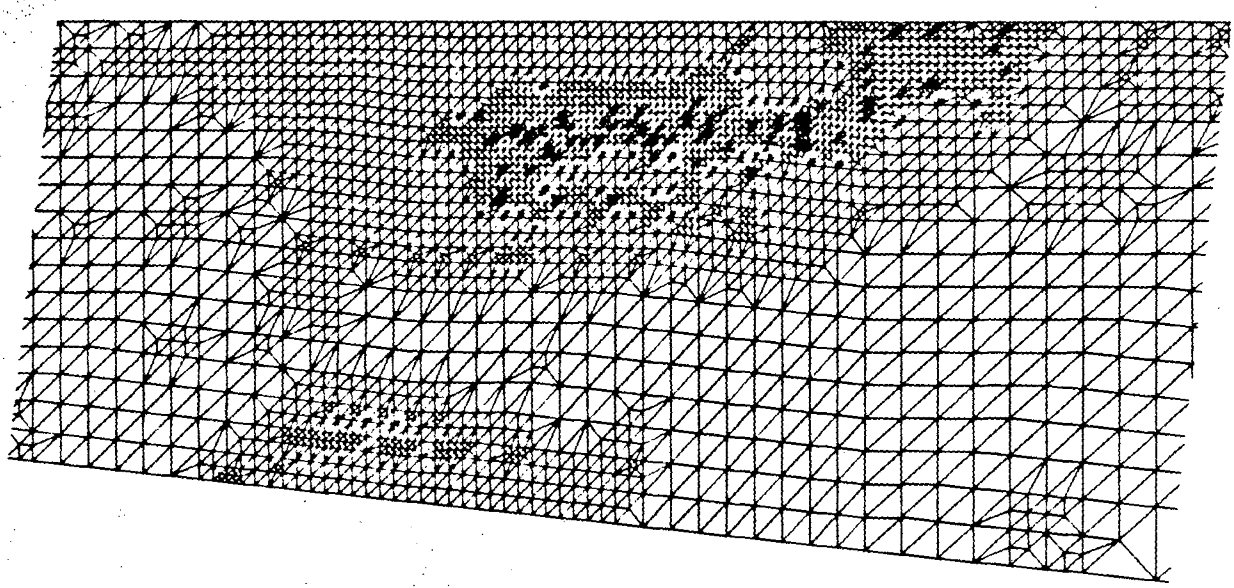


Figure 32: (con't) Triangulation for $\gamma = 30.6^o$, levels 4 and 5 magnified 5 times about the critical Γ.

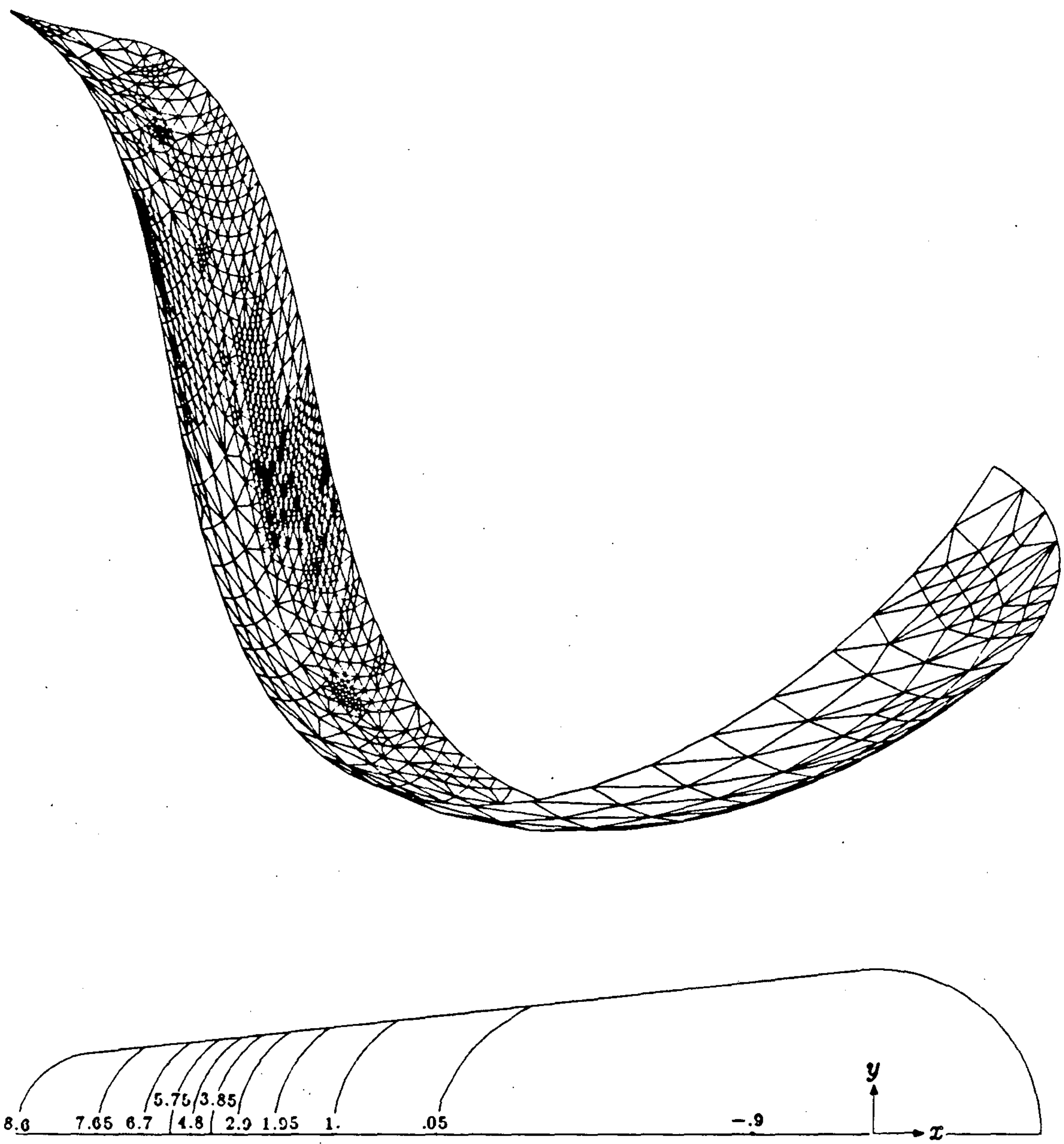


Figure 33: PLTMG surface and contour plots for $\gamma = 30.6^o$, level 4. Surface plot is the projection of the solution into the plane perpendicular to the vector $\vec{i} - \vec{j} - \vec{k}$. Contours of $u(x, y)$ are equally spaced between $-.9$ and 8.6, the minimum and maximum over the entire domain.

References

[1] Bank,R.and Rose,D., *Parameter selection for Newton-like methods applicable to non-linear differential equations*, SIAM J.Numer.Anal. **17** (1980), pp.806-822

[2] Bank,R., *A comparison of two multilevel iterative methods for nonsymmetric and indefinite elliptic finite element equations*, SIAM J.Numer.Anal. **18** (1981), pp.724-743

[3] Bank,R.,and Rose,D.,*Analysis of a multilevel iterative method for nonlinear finite element equations*, Math.Comp. **39** (1982),pp.453-465

[4] Bank,R., *PLTMG User's Guide*, ed.4.0, Dept.Math., Univ.Calif.San Diego, 1985

[5] Brown,R., *Finite element methods for the calculation of capillary surfaces*, J.Comp.Phys. **33** (1979), pp.217-235

[6] Concus,P.and Finn,R., *On capillary free surfaces in the absence of gravity*, Acta.Math. **132** (1974), pp.177-198

[7] Concus,P., *On existence criteria for fluid interfaces in the absence of gravity* in Waves on Fluid Surfaces, ed.R.Meyer, Academic Press, 1983, pp.113-122

[8] Concus,P.and Finn,R., *On the extremals of a subsidiary capillary problem*, J.reine angew.Math. **353** (1984), pp.215-220

[9] Concus,P., private communication, spring 1986

[10] Concus,P. and Finn,R., *Continuous and Discontinuous Disappearance of Capillary Surfaces*, to appear

[11] Finn,R., *Existence and non existence of capillary surfaces*, Manuscripta Math. **28** (1979), pp.1-11

[12] Finn,R.and Giusti,E., *Non existence and existence of capillary surfaces*, Manuscripta Math. **28** (1979), pp.13-20

[13] Finn,R., *Existence criteria for capillary free surfaces without gravity*, Indiana Univ.Math.J. **32** (1983), pp.439-460

[14] Finn,R., *A subsidiary variational problem and existence criteria for capillary surfaces*, J.reine angew.Math. **353** (1984), pp.196-214

[15] Giusti,E., *Boundary value problems for non-parametric surfaces of prescribed mean curvature*, Ann.Scuola Norm.Sup.Pisa Cl.Sci. **3** (1976), pp.501-548

[16] Laplace,P.de, Mécanique Céleste, Vol.IV, supp.to Book 10, trans.N.Bowditch, Little&Brown, Boston, 1839

[17] Roytburd,V., *On singularities of capillary surfaces in the absence of gravity*, Internat.J.Math.&Math.Sci. **6** (1983), pp.81-87

[18] Young,T., Miscellaneous Works, Vol.I, Ch.XIX-XXI, ed.G.Peacock, J.Murray, London, 1858

```
C------------------------------------------------------------------
C
C           PIECEWISE LINEAR TRIANGLE MULTI GRID PACKAGE
C
C                  EDITION 4.0 - - - MARCH, 1985
C
c    adapted to allow 'tacking down' of solution at first vertex
c                          Spring, 1986
c
C------------------------------------------------------------------
        INTEGER FUNCTION IDBC(I,IVERT)
            DIMENSION IVERT(2,1)
C
C       THIS LOOKS FOR DIRICHLET BOUNDARY POINTS
C       IBC=1  FOR DIRICHLET
C       IBC=0 OTHERWISE
C
        IDBC=0
c       This is the old line.
cbank   IF(IVERT(1,2)+IVERT(2,I).LT.0) IDBC=1
c       This is the new line.
        IF((I.eq.1).or.(IVERT(1,2)+IVERT(2,I).LT.0)) IDBC=1
        RETURN
        END
```

```
c-----------------------------------------------------------------

c                  Capillary Surface Problem

c            User-Supplied Functions and Subroutines for
c            Piecewise Linear Triangle Multi Grid Package

c                       for all domains

c-----------------------------------------------------------------

      function a1xy(x,y,u,ux,uy,rl,i,itype)
c     Evalutes a1 where div(a1,a2) = f  in OMEGA
c     and  (a1,a2) dot n = g2 on SIGMA2
c        u = u(x,y)
c        ux = du / dx
c        uy = du / dy
c        rl = continuation parameter
c        i = user triangle# s.t. (x,y) lies in closure of triangle i
c        itype = 1 -- a1
c              = 2 -- d(a1) / d(u)
c              = 3 -- d(a1) / d(ux)
c              = 4 -- d(a1) / d(uy)
c              = 5 -- d(a1) / d(rl)

      go to (10,20,30,40,50),itype
      write (6,1)
    1 format(' Warning:  invalid itype in a1xy.')
   10 continue
         a1xy = ux/sqrt(1.0e0 + ux*ux + uy*uy)
         return
   20 continue
         a1xy = 0.0e0
         return
   30 continue
         a1xy = (1.0e0 + uy*uy)/(sqrt(1.0e0 + ux*ux + uy*uy))**3
         return
   40 continue
         a1xy = -ux*uy/(sqrt(1.0e0 + ux*ux + uy*uy))**3
         return
   50 continue
         a1xy = 0.0e0
         return
      end

c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

      function a2xy(x,y,u,ux,uy,rl,i,itype)
c     Evalutes a2 where div(a1,a2) = f  in OMEGA
c     and  (a1,a2) dot n = g2 on SIGMA2
c        ux = du / dx
c        uy = du / dy
c        rl = continuation parameter
c        i = user triangle# s.t. (x,y) lies in closure of triangle i
```

```
c        itype = 1 -- a2
c              = 2 -- d(a2) / d(u)
c              = 3 -- d(a2) / d(ux)
c              = 4 -- d(a2) / d(uy)
c              = 5 -- d(a2) / d(rl)

      go to (10,20,30,40,50),itype
      write (6,1)
    1 format(' Warning:  invalid itype in a2xy.')
   10 continue
         a2xy = uy/sqrt(1.0e0 + ux*ux + uy*uy)
         return
   20 continue
         a2xy = 0.0e0
         return
   30 continue
         a2xy = -ux*uy/(sqrt(1.0e0 + ux*ux + uy*uy))**3
         return
   40 continue
         a2xy = (1.0e0 + ux*ux)/(sqrt(1.0e0 + ux*ux + uy*uy))**3
         return
   50 continue
         a2xy = 0.0e0
         return
      end

c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

      function fxy(x,y,u,ux,uy,rl,i,itype)
c     Evalutes f where div(a1,a2) = f  in OMEGA
c        u = u(x,y)
c        ux = du / dx
c        uy = du / dy
c        i = user triangle# s.t. (x,y) lies in closure of triangle i
c        itype = 1  -- f
c              = 2  -- d(f) / d(u)
c              = 3  -- d(f) / d(ux)
c              = 4  -- d(f) / d(uy)
c              = 5  -- d(f) / d(rl)

      common/user/bondno,bvcg,bvcgsq,const,cosgam,hv2,
     &            shift,shift2,twobma

      go to (10,20,30,40,50),itype
      write (6,1)
    1 format(' Warning:  invalid itype in fxy.')
   10 continue
         fxy = bondno*u + const
         return
   20 continue
         fxy = bondno
         return
   30 continue
         fxy = 0.0e0
```

```
          return
   40 continue
          fxy = 0.0e0
          return
   50 continue
          fxy = 0.0e0
          return
      end

c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

      function uxy(x,y,i,itype)
c     Evaluates the exact solution, if it is known.  This is used
c     in convergence studies.  The function here is the exact
c     solution for the circle of radius b with u(0,0) = 0.
c        i = user triangle# s.t. (x,y) lies in closure of triangle i
c        itype = 1 -- u
c              = 2 -- d(u) / d(x)
c              = 3 -- d(u) / d(y)

      common/user/bondno,bvcg,bvcgsq,const,cosgam,hv2,
     &            shift,shift2,twobma

      go to (10,20,30),itype
      write (6,1)
    1 format(' Warning:  invalid itype in uxy.')
   10 continue
         uxy = bvcg-sqrt(bvcgsq-x*x-y*y)
         return
   20 continue
         uxy = x/sqrt(bvcgsq-x*x-y*y)
         return
   30 continue
         uxy = y/sqrt(bvcgsq-x*x-y*y)
         return
      end

c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

      function pxy(x,y,u,ux,uy,rl,i,j,itype)
c     Evaluates integrands for triqud.
c        u = u(x,y)
c        ux = du / dx
c        uy = du / dy
c        rl = continuation parameter
c        i = user triangle# s.t. (x,y) lies on boundary of triangle i
c        j = edge number of triangle i
c        itype = 1 -- interior integrand
c              = 2 -- boundary integrand
c              = 3 -- boundary integrand for nx
c              = 4 -- boundary integrand for ny

      go to (10,20,30,40),itype
      write (6,1)
```

```
    1 format(' Warning:  invalid itype in pxy.')
   10 continue
         pxy = 1.0e0
         return
   20 continue
         pxy = 1.0e0
         return
   30 continue
         pxy = 0.0e0
         return
   40 continue
         pxy = 0.0e0
         return
      end

c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

      function qxy(x,y,u,ux,uy,rl,i)
c     Evaluates function for triplt.
c        u = u(x,y)
c        ux = du / dx
c        uy = du / dy
c        rl = continuation parameter
c        i = user triangle# s.t. (x,y) lies in closure of triangle i

      qxy = 0.0e0
      return
      end

c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

      subroutine usrcmd(list,llist)
      dimension list(1)

      return
      end

c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
c-----------------------------------------------------------------

c                     Capillary Surface Problem

c            User-Supplied Functions and Subroutines for
c            Piecewise Linear Triangle Multi Grid Package

c                      for circular domains
c              with axes of symmetry on x and y-axes
c              b = 1 = radius

c-----------------------------------------------------------------

      function gxy(x,y,u,rl,i,j,itype)
c     Evalutes g1 where  u = g1 on SIGMA1 (Dirichlet b.c.)
c     and g2 where  (a1,a2) dot n = g2 on SIGMA2 (natural b.c.)
c        u = u(x,y)
c        rl = continuation parameter
c        i = user triangle# s.t. (x,y) lies on boundary of triangle i
c        j = edge number of triangle i
c        itype = 1 -- g2
c              = 2 -- d(g2) / d(u)
c              = 3 -- d(g2) / d(rl)
c              = 4 -- g1
c              = 5 -- d(g1) / d(rl)
c              = 6 -- initial guess for nonlinear problem
c                     (If i=0, initial value for rl.  Otherwise,
c                      initial guess for the solution at the
c                      starting point for the continuation process.)

      common/user/bondno,bvcg,bvcgsq,const,cosgam,hv2,
     &            shift,shift2,twobma

      go to (10,20,30,40,50,60),itype
      write (6,1)
    1 format(' Warning:  invalid itype in gxy.')
   10 continue
        if (j.eq.1) then
c             On SIGMA.
              gxy = cosgam
           else
c             On axis of symmetry.
              gxy = 0.e0
           end if
        return
   20 continue
        gxy = 0.e0
        return
   30 continue
        gxy = 0.e0
        return
   40 continue
        gxy = 0.e0
        return
```

```
   50 continue
         gxy = 0.e0
         return
   60 continue
         if (i.ne.0) then
c              Zero initial guess.
               gxy = 0.e0
c              Exact initial guess.
c              gxy = bvcg - sqrt(bvcgsq - x*x - y*y)
            else
               gxy = 0.e0
            end if
         return
      end

c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

      subroutine gdata
c     Requests dimensions, bond number, and triangulation type.
c     Calculates variables in common area labeled user.
c     Fills vectors of starting coordinates and matrices of
c     starting triangle specifications.

      parameter(MIC=  500,MIV=  2000,MIT=  4000,LENW=  50000)

      common/verts/nv,vx(MIV),vy(MIV),lxy(MIV)
      common/tris/nt,itnode(3,MIT),itedge(3,MIT)
      common/mdpts/nc,xm(MIC),ym(MIC)
      common/rgns/nr,ib(51),jb(500),isym(50)
      common ip(100),w(LENW)
      common/user/bondno,bvcg,bvcgsq,const,cosgam,hv2,
     &            shift,shift2,twobma

c     Data for triangulation type = 0 or 1.
      data itnode(1,3),itnode(2,3),itnode(3,3)/1,5,3/
      data itnode(1,4),itnode(2,4),itnode(3,4)/1,3,6/
      data itedge(1,3),itedge(2,3),itedge(3,3)/0,0,1/
      data itedge(1,4),itedge(2,4),itedge(3,4)/0,1,0/
      data vx(6),vy(5)/0.e0,0.e0/

      b = 1.e0
      pi = 3.141592653589793

c     Set iprob = 6 for no continuation.
      ip(4) = 6

c     Request dimensions, bond number, and triangulation type.
      write(6,160)
  160 format(' Enter 0. < gamma < 90. degrees')
      read (5,120) gamma
  120 format(f15.10)
      write(6,180)
  180 format(' Enter 0. <= bond number')
      read (5,120) bondno
```

```
      write (6,190)
  190 format(' Enter 0 for 2 slices cut right'/
     &       '       1 for 2 slices cut right and top'/
     &       '       2 <= nelem for nelem pie slices')
      read (5,192) itri
  192 format(i2)
      if ((itri.eq.0).or.(itri.eq.1)) then
            nelem = 2
         else if (itri.ge.2) then
            nelem = itri
            itri = 2
         else
            write (6,197)
  197       format(' Warning: invalid itri in gdata.')
         end if

c     Calculate SIGMA, etc.
      SIGMA = 2.e0*pi*b
      OMEGA = pi*b*b
      SIGvOM = 2.e0/b
      cosgam = cos(gamma*pi/180.e0)
      const = (SIGMA*cosgam - bondno)/OMEGA
      bvcg = b/cosgam
      bvcgsq = bvcg*bvcg

c     nc = number of curved edges+1
      nc = nelem + 1
c     nr = number of regions
      nr = 1
c     nt = number of triangles
      nt = nelem
c     nv = number of vertices
      nv = nelem + 2

c     Fill (vx(i),vy(i)) = (xcoord,ycoord) of vertex #i, i=1,nv
      piv2nl = .5e0*pi/nelem
      vx(1) = 0.e0
      vy(1) = 0.e0
      do 200 i = 2,nv
         arg =  (i-2.e0)*piv2nl
         vx(i) = b*cos(arg)
         vy(i) = b*sin(arg)
  200    continue

c     Fill itnode(1to3,i) = vertex numbers of triangle i, i=1,nt
c                           +k natural curved edge k
c                           +1 natural straight edge
c     Fill itedge(1to3,i) = 0 internal edge          ,i=1,nt
c                           -1 Dirichlet straight edge
c                           -k Dirichlet curved edge k
      do 400 i = 1,nt
         itnode(1,i) = 1
         itnode(2,i) = i + 1
         itnode(3,i) = i + 2
```

```
          itedge(1,i) = i + 1
          itedge(2,i) = 0
          itedge(3,i) = 0
 400      continue
       itedge(3,1) = 1
       itedge(2,nt) = 1
       itrip1 = itri + 1
       go to (520,510,530),itrip1
       write(6,505)
  505 format(' Warning invalid itrip1 in gdata')
  510 continue
          itnode(1,2) = 6
          vy(6) = vy(3)
  520 continue
          nt = nt + itrip1
          nv = nv + itrip1
          itnode(1,1) = 5
          vx(5) = vx(3)
  530 continue

c      Fill (xm(i),ym(i)) = (xcoord,ycoord) of midpoint #i, i=2,nc
       xm(1) = 0.e0
       ym(1) = 0.e0
       do 700 i = 2,nc
         arg = (i-1.5e0)*piv2nl
         xm(i) = b*cos(arg)
         ym(i) = b*sin(arg)
  700    continue

c      Set for adaptive refinement only.
       do 800 i = 1,nv
          lxy(i) = 1
  800     continue

c      write(6,910) (vx(k),k=1,nv)
c 910 format(' vx',10(1x,f6.3))
c      write(6,920) (vy(k),k=1,nv)
c 920 format(' vy',10(1x,f6.3))
c      write(6,930) (xm(k),k=1,nc)
c 930 format(' xm',10(1x,f6.3))
c      write(6,940) (ym(k),k=1,nc)
c 940 format(' ym',10(1x,f6.3))
c      do 965 i = 1,3
c         write(6,960) i,(itnode(i,k),k=1,nt)
c 960     format(' itnode(',i1,',)',20(1x,i2))
c 965     continue
c      do 975 i = 1,3
c         write(6,970) i,(itedge(i,k),k=1,nt)
c 970     format(' itedge(',i1,',)',20(1x,i2))
c 975     continue
c      write(6,990) b,gamma,bondno,SIGvOM
c 990 format('      b         gamma       bondno  SIGMA/OMEGA'/
c     &         4(1x,f10.6)/)
```

```
return
end
```

```
c-----------------------------------------------------------------

c                       Capillary Surface Problem

c            User-Supplied Functions and Subroutines for
c            Piecewise Linear Triangle Multi Grid Package

c                        for trapezoidal domains
c                  with axis of symmetry on y-axis, base on x-axis
c                  h = height
c                  b = 1 = half length of base
c                  a = half length of top < b

c-----------------------------------------------------------------

      function gxy(x,y,u,rl,i,j,itype)
c     Evalutes g1 where  u = g1 on SIGMA1 (Dirichlet b.c.)
c     and g2 where  (a1,a2) dot n = g2 on SIGMA2 (natural b.c.)
c        u = u(x,y)
c        rl = continuation parameter
c        i = user triangle# s.t. (x,y) lies on boundary of triangle i
c        j = edge number of triangle i
c        itype = 1 -- g2
c              = 2 -- d(g2) / d(u)
c              = 3 -- d(g2) / d(rl)
c              = 4 -- g1
c              = 5 -- d(g1) / d(rl)
c              = 6 -- initial guess for nonlinear problem
c                     (If i=0, initial value for rl.  Otherwise,
c                      initial guess for the solution at the
c                      starting point for the continuation process.)

      common/user/bondno,bvcg,bvcgsq,const,cosgam,hv2,
     &            shift,shift2,twobma

      go to (10,20,30,40,50,60),itype
      write (6,1)
    1 format(' Warning:  invalid itype in gxy.')
   10 continue
         if (j.ne.2) then
c              On SIGMA.
               gxy = cosgam
            else
c              On axis of symmetry.
               gxy = 0.0e0
            end if
         return
   20 continue
         gxy = 0.0e0
         return
   30 continue
         gxy = 0.0e0
         return
   40 continue
```

```
            gxy = 0.0e0
            return
   50    continue
            gxy = 0.0e0
            return
   60    continue
         if (i.ne.0) then
c                Zero initial guess.
c                gxy = 0.0e0
c                Initial guess based on the solution for the circle.
                 gxy = (bvcg - sqrt(bvcgsq-x*x-shift2*(y-hv2)**2))
     &                   /shift  + y*twobma
c                write(10,100) x,y,gxy
c100             format(25x,2(3x,e14.7),3x,e17.10)
            else
               gxy = 0.e0
            end if
C
            return
         end

c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

         subroutine gdata
c        Requests dimensions, bond number, and triangulation type.
c        Calculates variables in common area labeled user.
c        Fills vectors of starting coordinates and matrices of
c        starting triangle specifications.

         parameter(MXC=  500,MXV=  2000,MXT=  4000,LENW=  50000)

         common/verts/nv,vx(MXV),vy(MXV),lxy(MXV)
         common/tris/nt,itnode(3,MXT),itedge(3,MXT)
         common/mdpts/nc,xm(MXC),ym(MXC)
         common/rgns/nr,ib(51),jb(500),isym(50)
         common ip(100),w(LENW)
         common/user/bondno,bvcg,bvcgsq,const,cosgam,hv2,
     &              shift,shift2,twobma

         b = 1.e0
         pi = 3.141592653689793
         piv180 = pi/180.e0

c        Set iprob = 6 for no continuation.
         ip(4) = 6

c        Request dimensions, bond number, and triangulation type.
         write(6,100) b
  100    format(' Enter 0.< a < ',f10.5)
         read (5,120) a
  120    format(f15.10)
         write(6,140)
  140    format(' Enter 0. < h')
         read (5,120) h
```

```
      write(6,160)
  160 format(' Enter 0. < gamma < 90. degrees')
      read (5,120) gamma
      write(6,180)
  180 format(' Enter 0. <= bond number')
      read (5,120) bondno
      write(6,195)
  195 format(' Enter 1<=nyel for nyel evenly-spaced elements'
     &       ' on y-axis,'/
     &       '          -nyel for nyel unevenly-spaced elements')
      read(5,198) nyel
  198 format(i3)
      if (nyel.ge.1) then
            itri = 1
         else if (nyel.le.-1) then
            itri = 0
            nyel = -nyel
         else
            write(6,199)
  199       format(' Warning: invalid nyel in gdata.')
         end if

c     Calculate SIGMA, etc.
      bma = b - a
      apb = a + b
      twobma = 2.e0*bma
      SIGMA =  2.e0*(apb + sqrt(h*h + bma*bma))
      OMEGA = h*(apb)
      SIGvOM = SIGMA/OMEGA
      cosgam = cos(gamma*piv180)
      R = 1.e0/(SIGvOM*cosgam)
      const = (SIGMA*cosgam - bondno)/OMEGA
      bvcgsq = (b/cosgam)**2
      bvcg = sqrt(bvcgsq - b*b)
      hv2 = 0.5e0*h
      shift = 2.e0*b/h
      shift2 = shift*shift
      hvnyel = h/nyel
      if (bma.ne.0.) then
            alpha = atan(h/bma)
            amg = alpha - gamma*piv180
            Rcntr = h - R*cos(amg)
     &            - (R*sin(amg)-a)*tan(alpha)
         else
            alpha = .5e0*pi
c           Rcntr is not applicable if SIGMA = rectangle.
            Rcntr = 0.
         end if

c     nc = number of curved edges+1
      nc = nelem + 1
c     nr = number of regions
      nr = 1
c     nt = number of triangles
```

```
      nt = 2*nyel
c     nv = number of vertices
      nv = 2*nyel + 2

c     Fill (vx(i),vy(i)) = (xcoord,ycoord) of vertex #i, i=1,nv
      if (itri.ne.1) then
           vy( 1) = 0.00e0
           do 275 k = 1,nyel-1
              konst = 2*k + 1
              write(6,270) konst
 270          format(' vy(',i2,')=______*h?')
              read(5,120) blank
              vy(konst) = blank*h
 275          continue
           vy(nv-1) = h
        end if
        do 200 i = 0,nyel
           ieven = 2*i + 2
           iodd = 2*i + 1
           if (itri.eq.1) vy(iodd) = i*hvnyel
           vy(ieven) = vy(iodd)
           vx(iodd) = 0.e0
           vx(ieven) = a + bma*(h-vy(iodd))/h
 200       continue

c     Fill itnode(1to3,i) = vertex numbers of triangle i, i=1,nt
c                         +k natural curved edge k
c                         +1 natural straight edge
c     Fill itedge(1to3,i) = 0 internal edge          ,i=1,nt
c                          -1 Dirichlet straight edge
c                          -k Dirichlet curved edge k
        do 400 i = 1,nyel
           ix2 = i*2
           ix2m1 = ix2 - 1
           ix2p2 = ix2 + 2
           itnode(1,ix2m1) = ix2m1
           itnode(2,ix2m1) = ix2p2
           itnode(3,ix2m1) = ix2 + 1
           itnode(1,ix2) = ix2m1
           itnode(2,ix2) = ix2
           itnode(3,ix2) = ix2p2
           itedge(1,ix2m1) = 0
           itedge(2,ix2m1) = 1
           itedge(3,ix2m1) = 0
           itedge(1,ix2) = 1
           itedge(2,ix2) = 0
           itedge(3,ix2) = 0
 400       continue
        itedge(3,2) = 1
        itedge(1,nt-1) = 1

c     Fill (xm(i),ym(i)) = (xcoord,ycoord) of midpoint #i, i=2,nc
      xm(1) = 0.e0
      ym(1) = 0.e0
```

```
c     Set for adaptive refinement only.
      do 800 i = 1,nv
         lxy(i) = 1
  800    continue

c     write(6,910) (vx(k),k=1,nv)
c 910 format(' vx',10(1x,f6.3))
c     write(6,920) (vy(k),k=1,nv)
c 920 format(' vy',10(1x,f6.3))
c     write(6,930) (xm(k),k=1,nc)
c 930 format(' xm',10(1x,f6.3))
c     write(6,940) (ym(k),k=1,nc)
c 940 format(' ym',10(1x,f6.3))
c     do 965 i = 1,3
c        write(6,960) i,(itnode(i,k),k=1,nt)
c 960    format(' itnode(',i1,',)',20(1x,i2))
c 965    continue
c     do 975 i = 1,3
c        write(6,970) i,(itedge(i,k),k=1,nt)
c 970    format(' itedge(',i1,',)',20(1x,i2))
c 975    continue
c     write(6,990) b,a,h,gamma,bondno,SIGvOM,R,Rcntr
c 990 format('      b         a         h        gamma   ',
c    &        ' bondno SIGMA/OMEGA     R      Rcntr'
c    &/8(1x,f9.6)/)

      return
      end
```

```
c-----------------------------------------------------------------

c                    Capillary Surface Problem

c          User-Supplied Functions and Subroutines for
c          Piecewise Linear Triangle Multi Grid Package

c               for curvilinear trapezoidal domains
c                 with axis of symmetry on x-axis
c                 capl = length of axis of symmetry
c                 b = 1 = radius of larger circle
c                 a = radius of smaller circle < b

c-----------------------------------------------------------------

      function gxy(x,y,u,rl,i,j,itype)
c     Evalutes g1 where  u = g1 on SIGMA1 (Dirichlet b.c.)
c     and g2 where  (a1,a2) dot n = g2 on SIGMA2 (natural b.c.)
c        u = u(x,y)
c        rl = continuation parameter
c        i = user triangle# s.t. (x,y) lies on boundary of triangle i
c        j = edge number of triangle i
c        itype = 1 -- g2
c              = 2 -- d(g2) / d(u)
c              = 3 -- d(g2) / d(rl)
c              = 4 -- g1
c              = 5 -- d(g1) / d(rl)
c              = 6 -- initial guess for nonlinear problem
c                     (If i=0, initial value for rl.  Otherwise,
c                      initial guess for the solution at the
c                      starting point for the continuation process.)

      common/user/bondno,bvcg,bvcgsq,const,cosgam,hv2,
     &            shift,shift2,twobma

      go to (10,20,30,40,50,60),itype
      write (6,1)
    1 format(' Warning:  invalid itype in gxy.')
   10 continue
         if (j.ne.2) then
c              On SIGMA.
               gxy = cosgam
            else
c              On axis of symmetry.
               gxy = 0.0e0
            end if
         return
   20 continue
         gxy = 0.0e0
         return
   30 continue
         gxy = 0.0e0
         return
   40 continue
```

```
         gxy = 0.0e0
         return
  50  continue
         gxy = 0.0e0
         return
  60  continue
      if (i.ne.0) then
c           Zero initial guess.
c           gxy = 0.0e0
c           Initial guess based on the solution for the circle.
            if (x.lt.0.) then
                  gxy = (bvcg - sqrt(bvcgsq-shift2*x*x-y*y))
     &                  - x*cosgam
               else
                  gxy = (bvcg - sqrt(bvcgsq-x*x-y*y))
               end if
c           write(10,100) x,y,gxy
c100        format(25x,2(3x,e14.7),3x,e17.10)
         else
            gxy = 0.e0
         end if
C
         return
      end

c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

      subroutine gdata
c     Requests dimensions, bond number, and triangulation type.
c     Calculates variables in common area labeled user.
c     Fills vectors of starting coordinates and matrices of
c     starting triangle specifications.

      parameter(MXC=  500,MXV=  2000,MXT=  4000,LENW=  50000)

      common /verts/nv,vx(MXV),vy(MXV),lxy(MXV)
      common /tris/nt,itnode(3,MXT),itedge(3,MXT)
      common /mdpts/nc,xm(MXC),ym(MXC)
      common /rgns/nr,ib(51),jb(500),isym(50)
      common ip(100),w(LENW)
      common/user/bondno,bvcg,bvcgsq,const,cosgam,hv2,
     &            shift,shift2,twobma

      b = 1.e0
      pi = 3.141592653689793
      piv180 = pi/180.e0

c     Set iprob = 6 for no continuation.
      ip(4) = 6

c     Request dimensions, bond number, and triangulation type.
      write(6,110) 2.e0*b
  110 format(' Enter ',f10.5,' < L .')
      read (5,120) capl
```

```
  120 format(f15.10)
      write(6,140) b
  140 format(' Enter 0. < a < ',f10.5)
      read (5,120) a
      write(6,160)
  160 format(' Enter 0. < gamma < 90. degrees.')
      read (5,120) gamma
      write(6,180)
  180 format(' Enter 0. <= bond number')
      read (5,120) bondno
      write(6,195)
  195 format(' Enter 1<=nxel for nxel evenly-spaced elements'
     &       ' on x-axis between 0. and acntr,'/
     &       '          -nxel for nxel unevenly-spaced elements.')
      read(5,198) nxel
  198 format(i3)
      if (nxel.ge.1) then
            itri = 1
         else if (nxel.le.-1) then
            itri = 0
            nxel = -nxel
         else
            write(6,197)
  197       format(' Warning: invalid nxel in gdata.')
         end if

c     Calculate SIGMA, etc.
      bma = b - a
      phi = acos(bma/(capl-a-b))
      alpha = 90.e0 - phi/piv180
      cosphi = cos(phi)
      sinphi = sin(phi)
      const = tan(phi) - phi
      SIGMA = 2.e0*((const+pi)*b   - const*a  )
      OMEGA =      ((const+pi)*b*b - const*a*a)
      SIGvOM = SIGMA/OMEGA
      cosgam = cos(gamma*piv180)
      const = (SIGMA*cosgam - bondno)/OMEGA
      bvcgsq = (b/cosgam)**2
      bvcg = sqrt(bvcgsq -  b*b)
      acntr = -(capl - a - b)
      R = 1./(SIGvOM*cosgam)
      Rcntr = (R*cosgam -b)/cosphi
      nxelx2 = 2*nxel
      shift = -b/(b-capl)
      shift2 = shift*shift

c     nc = number of curved edges+1
      nc = 4
c     nr = number of regions
      nr = 1
c     nt = number of triangles
      nt = 3 + nxelx2
c     nv = number of vertices
```

```
      nv = 5 + nxelx2

c     Fill (vx(i),vy(i)) = (xcoord,ycoord) of vertex #i, i=1,nv
      vx( 1) =  b
      vy( 1) =  0.e0
      pimphi = pi - phi
      arg = 0.5e0*pimphi
      vx( 2) =  b*cos(arg)
      vy( 2) =  b*sin(arg)
      xelem = acntr/nxel
      if(itri.ne.1) then
            vx( 3) = 0.e0
            do 275 k = 1,nxel-1
               konst = 2*k + 3
               write(6,270) konst
  270          format(' vx(',i2,')=______*acntr?.')
               read(5,120) blank
               vx(konst) = blank*acntr
  275          continue
            vx(nv-2) = acntr
         end if
      do 300 i = 0,nxel
         ieven = 2*i + 4
         iodd = ieven - 1
         if (itri.eq.1) vx(iodd) = i*xelem
         vy(iodd) = 0.e0
         temp = a + (vx(iodd)-acntr)*cosphi
         vx(ieven) = vx(iodd) - temp*cosphi
         vy(ieven) = temp*sinphi
  300    continue
       vx(nv) = b - capl
       vy(nv) = 0.

c     Fill itnode(1to3,i) = vertex numbers of triangle i, i=1,nt
c                           +k natural curved edge k
c                           +1 natural straight edge
c     Fill itedge(1to3,i) = 0 internal edge          ,i=1,nt
c                           -1 Dirichlet straight edge
c                           -k Dirichlet curved edge k
        do 400 i = 1,nxel+2
           ix2 = 2*i
           ix2m1 = ix2 - 1
           ix2p1 = ix2 + 1
           itnode(1,ix2) = ix2p1
           itnode(2,ix2) = ix2
           itnode(3,ix2) = ix2 + 2
           itnode(1,ix2m1) = ix2m1
           itnode(2,ix2m1) = ix2
           itnode(3,ix2m1) = ix2p1
           itedge(1,ix2) = 1
           itedge(2,ix2) = 0
           itedge(3,ix2) = 0
           itedge(1,ix2m1) = 0
           itedge(2,ix2m1) = 1
```

```
            itedge(3,ix2m1) = 0
  400       continue
         itedge(3,1) = 2
         itedge(1,2) = 3
         itedge(1,nt) = 4


c     Fill (xm(i),ym(i)) = (xcoord,ycoord) of midpoint #i, i=2,nc
      xm( 1) =  0.e0
      ym( 1) =  0.e0
      arg = 0.25e0*pimphi
      xm( 2) =  b*cos(arg)
      ym( 2) =  b*sin(arg)
      arg = 0.75e0*pimphi
      xm( 3) =  b*cos(arg)
      ym( 3) =  b*sin(arg)
      arg = .5e0*phi
      xm( 4) =  acntr - a*cos(arg)
      ym( 4) =  a*sin(arg)

c     Set for adaptive refinement only.
      do 800 i = 1,nv
         lxy(i) = 1
  800    continue

c     write(6,910) (vx(k),k=1,nv)
c 910 format(' vx',10(1x,f6.3))
c     write(6,920) (vy(k),k=1,nv)
c 920 format(' vy',10(1x,f6.3))
c     write(6,930) (xm(k),k=1,nc)
c 930 format(' xm',10(1x,f6.3))
c     write(6,940) (ym(k),k=1,nc)
c 940 format(' ym',10(1x,f6.3))
c     do 965 i = 1,3
c        write(6,960) i,(itnode(i,k),k=1,nt)
c 960    format(' itnode(',i1,',)',20(1x,i2))
c 965    continue
c     do 975 i = 1,3
c        write(6,970) i,(itedge(i,k),k=1,nt)
c 970    format(' itedge(',i1,',)',20(1x,i2))
c 975    continue
c     write(6,990) b,capl,a,alpha,gamma,bondno,SIGvOM,R,Rcntr
c 990 format('     b         L         a      alpha     gamma   ',
c    &' bondno SIGMA/OMEGA   R       Rcntr'/9(1x,f8.5)/)

      return
      end
```