

A FREE BOUNDARY PROBLEM FOR CAPILLARY SURFACES ARISING
FROM THE EQUILIBRIUM OF A FLOATING DROP.

A Thesis by

RAY TREINEN

B.S., WICHITA STATE UNIVERSITY, 1999

Submitted to the Department of Mathematics and Statistics and the faculty of the
Graduate School of Wichita State University in partial fulfillment of the
requirements for the degree of Master of Science

Spring 2001

A free boundary problem for capillary surfaces arising from the equilibrium of a
floating drop

I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major Mathematics.

Dr. Alan Elcrat, Committee Chair

We have read this thesis
and recommend its acceptance:

Dr. Vassilis Papanicolaou, Committee Member

Dr. Mel Zandler, Committee Member

Acknowledgements

I would like to thank Dr. Alan Elcrat for his help and guidance with this work. His ideas always lead me to a better way of doing things. I would also like to thank Dr. Vassilis Papanicolaou and Dr. Mel Zandler for being on the committee. I am also indebted to Dr. Tomasz Hrycak for his timely assistance and encouragement and to Kevin Hemphill with whom I began this project. This research was supported by the NSF under Cooperative Agreement EPS-9874732. I am also grateful for funding from the Dora Wallace Hodgson Graduate Research Award.

Abstract

A drop of fluid rests on a another fluid. We assume that it is axisymmetric. We solve this problem at a given radius using a system of differential equations (capillary surface equations) for each surface u , v , and w and making the discrepancy between them zero. This is achieved numerically using Matlab.

Contents

1	Introduction	1
2	The Sessile Drop	3
3	The Free Boundary Problem	6
4	Results	10
5	Appendix	16

List of Figures

1	$r = 1, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20$ and $\sigma_{01} = 10, \sigma_{02} = 10, \sigma_{12} = 15$. . .	1
2	A typical sessile surface, with $\bar{\psi} = \pi$ here.	3
3	A typical outer surface with ψ at the top corresponding to $-\pi$. . .	4
4	The sessile surface that we calculate, ending at \bar{r}	5
5	The angles between the surfaces.	6
6	A typical iteration while varying $\bar{\psi}$	9
7	A converged case before we extend the outer surface.	9
8	$\bar{r} = 1, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 10, \sigma_{02} = 20, \sigma_{12} = 30$	10
9	$\bar{r} = .5, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 10, \sigma_{02} = 20, \sigma_{12} = 30$	11
10	$\bar{r} = .3, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 10, \sigma_{02} = 15, \sigma_{12} = 20$	11
11	$\bar{r} = .28, \rho_0 = 0, \rho_1 = 7, \rho_2 = 20, \sigma_{01} = 12, \sigma_{02} = 13, \sigma_{12} = 23$	11
12	$\bar{r} = 1, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 25, \sigma_{02} = 13, \sigma_{12} = 12$	11
13	$\bar{r} = 1, \rho_0 = 0, \rho_1 = 15, \rho_2 = 20, \sigma_{01} = 25, \sigma_{02} = 13, \sigma_{12} = 12$	11
14	$\bar{r} = 1, \rho_0 = 0, \rho_1 = 17, \rho_2 = 20, \sigma_{01} = 25, \sigma_{02} = 13, \sigma_{12} = 12$	11
15	$\bar{r} = 1, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 15, \sigma_{02} = 15, \sigma_{12} = 20$	11
16	$\bar{r} = 1, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 15, \sigma_{02} = 13, \sigma_{12} = 12$	11
17	$\bar{r} = 1, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 19.5, \sigma_{02} = 15, \sigma_{12} = 20$	11
18	$\bar{r} = 1, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 12, \sigma_{02} = 13, \sigma_{12} = 20$	11
19	$\bar{r} = 1.1, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 12, \sigma_{02} = 13, \sigma_{12} = 20$	11
20	$\bar{r} = 1.25, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 12, \sigma_{02} = 13, \sigma_{12} = 20$	12

List of Tables

1	Data from various configurations of \bar{r} , ρ , and σ	13
---	---	----

List of Symbols

g	acceleration due to gravity
r	the distance from the vertical axis: radius
u	height of the upper surface
v	height of the lower surface
w	height of the outer surface
dS	infinitesimal surface area
dV	infinitesimal volume of the liquid
F	difference between $u(\bar{r})$ and $w(\bar{r})$
H	mean curvature
U	height of a sessile surface
V	volume of the liquid, later height of a sessile surface
k_l	latitudinal curvature
k_m	meridinal curvature
\bar{r}	given radius
\dot{r}	derivative of r with respect to u_0
$r_{-\frac{\pi}{2}}$	radius of vertical point of w
u_0	value of u at $r = 0$

- \dot{u} derivative of u with respect to u_0
- $u_{-\frac{\pi}{2}}$ height of vertical point of w
- κ capillary constant ($\frac{\rho g}{\sigma}$)
- λ Lagrange multiplier from volume constraint
- ψ tangent angle of each surface
- ρ density difference of fluids
- σ surface tension
- $\beta\sigma$ proportionality constant of adhesion
- γ_{01} angle between v and w
- γ_{02} angle between u and v
- γ_{12} angle between u and w
- κ_{01} capillary constant for u
- κ_{02} capillary constant for w
- κ_{12} capillary constant for v
- $\bar{\psi}$ tangent angle at \bar{r} generally for v
- $\bar{\psi}_u$ tangent angle at \bar{r} of u
- $\bar{\psi}_w$ tangent angle at \bar{r} of w
- ρ_0 density of the fluid above
- ρ_1 density of the fluid inside the drop

ρ_2	density of the fluid below
σ_{01}	surface tension in the direction of u
σ_{02}	surface tension in the direction of w
σ_{12}	surface tension in the direction of v
\mathcal{E}	total energy of the system
$\delta\mathcal{E}$	variation of energy
Λ	interface between gas and liquid
Λ^*	interface between liquid and container
Ω	domain considered
Υ_ρ	potential energy

1 Introduction

On an infinite sheet of fluid rests a drop. We assume this drop to be axisymmetric. Each fluid has a density: ρ_0, ρ_1, ρ_2 with $\rho_0 < \rho_1 < \rho_2$. A typical example would be a drop of oil resting on water with air above. Also given are the radius and the surface tensions: $\sigma_{01}, \sigma_{12}, \sigma_{02}$ where each of these must be less than the sum of the other two to achieve a force balance.

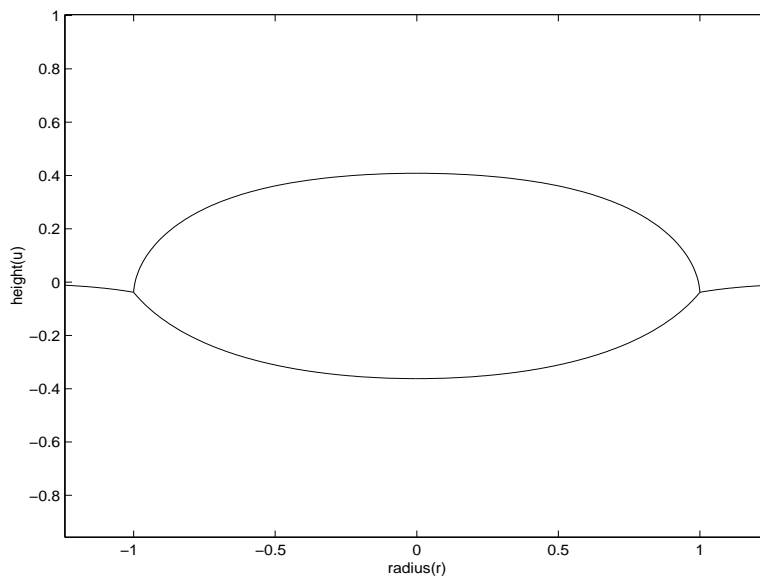


Figure 1: $r = 1, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20$ and $\sigma_{01} = 10, \sigma_{02} = 10, \sigma_{12} = 15$.

To justify the approach we take, we discuss the energy of the system [3]. A system of a bounded container containing a liquid, and a gas, we write the total energy of the system as

$$\mathcal{E} = \sigma \int \int_{\Lambda} dS + \beta\sigma \int \int_{\Lambda^*} dS + \int \int \int_{\Omega} \Upsilon_{\rho} dV + \lambda V. \quad (1)$$

We define Λ to be the liquid/gas interface, Λ^* , the interface between the liquid and the container, where we have the proportionality constant $\beta\sigma$ for the adhesion there. Further we have Υ_ρ , the gravitational energy per unit volume for each liquid of density ρ . The first two terms of the functional are due to energy from surface effects. The third term is the potential energy due to gravity, and the last term is a constraint condition on the volume of the liquid, where λ is the Lagrange multiplier.

The idea here is to perturb the surface Λ and calculate the variation of energy $\delta\mathcal{E}$, and setting it equal to zero. We have [2]

$$2H = \frac{\Upsilon_\rho}{\sigma} + \lambda, \quad (2)$$

where H is the mean curvature and with this we have the boundary condition $\cos\gamma = \beta$ where the surface meets the container. So this drop minimizes the potential energy of the system, and each surface is determined by the equivalent equation

$$\text{div}Tu = \kappa u + \lambda \quad (3)$$

where

$$Tu = \frac{\nabla u}{\sqrt{1 + |\nabla u|^2}}. \quad (4)$$

Here each surface has different boundary conditions, and we will delve into the details of that in Chapter 3, but for now it suffices to say that they only depend on the surface tensions in the three phase problem. We use the axisymmetric

systems of ODE's parametrized by contact angle for ease of computations [2].

Thus we have

$$\frac{dr}{d\psi} = \frac{r \cos \psi}{\kappa r u - \sin \psi}, \quad \frac{dr}{d\psi} = \frac{r \sin \psi}{\kappa r u - \sin \psi} \quad (5)$$

for each surface, with the boundary conditions as just mentioned. Thus we are not restricting ourselves to non parametric surfaces; since we parametrize with respect to contact angle we are allowing our surfaces to fold over.

2 The Sessile Drop

For the upper and lower surfaces we need to construct sessile surfaces. Moreover, we need these surfaces to match boundary conditions that have $\psi = 0$ and $u = u_0$ at $r = 0$, while at the prescribed radius, \bar{r} , $\bar{\psi}$ is given. First we state a lemma [2]:

Lemma 2.1 *On the curve $u(\psi; u_0)$, k_m increases over the entire traverse $0 < \psi < \pi$; k_l increases until the vertical point $\psi = \frac{\pi}{2}$, and then decreases until $\psi = \pi$.*

Where k_m and k_l are defined to be the meridinal and latitudinal curvatures. Now the fact that k_m , and thus ψ , is increasing throughout the interval is important in what follows. We use the parametric representation of these differential equations (5) to begin with a height u_0 at $r = 0$ that we use as initial conditions in the MATLAB function ODE45 to integrate from $\psi = 0$ to $\bar{\psi}$ and find the value of r at which we have $\bar{\psi}$. The differential equations are undefined for $\psi = 0$ and $r = 0$. We use l'Hopital's rule to derive

$$\frac{dr}{d\psi} = \frac{2}{\kappa u_0}, \quad \frac{du}{d\psi} = 0 \quad (6)$$

for $\psi \rightarrow 0$. The lemma states that there is only one value of $\psi = \bar{\psi}$ for any particular u_0 , and thus this method of finding a surface given u_0 is justified. From the proof of Theorem 3.2 in Finn [2], we have the result that $\frac{d\bar{u}}{d\bar{\psi}} > 0$ and $\frac{d\bar{r}}{d\bar{\psi}} < 0$, where $\bar{\cdot}$ denotes the derivative with respect to the initial height,

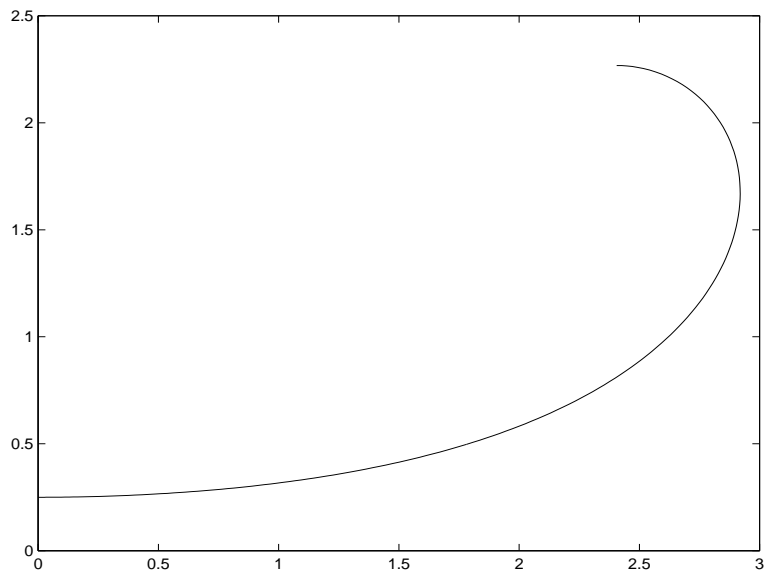


Figure 2: A typical sessile surface, with $\bar{\psi} = \pi$ here.

u_0 . This implies the uniqueness of each sessile drop, with respect to the initial height u_0 , and hope to our being able to numerically find a surface for each of these values of u_0 . Now we use these ideas inside the MATLAB function FZERO. This function varies the initial value of u_0 finding the particular u_0 that has $\psi = \bar{\psi}$ at r and $r = \bar{r}$. Thus we have generated a sessile surface meeting all the boundary conditions that we need.

Now we look at the outer surface. It uses the same differential equations as the inner surfaces, but it begins with a different set of boundary conditions. Of course the condition that as $r \rightarrow \infty$, $u \rightarrow 0$ is needed. We want to parametrize with respect to ψ , so we adapt the standard usage of the differential equations for the sessile drop to fit our needs, using ideas of Vogel [6]. We start out at

$\psi = -\pi$ where the surface is horizontal at some finite radius, and then move along through $\psi = -\frac{\pi}{2}$ where the surface is vertical, to $\psi = 0$ where $r \rightarrow \infty$ and $u \rightarrow 0$. So in numerically constructing this surface we need some sort of

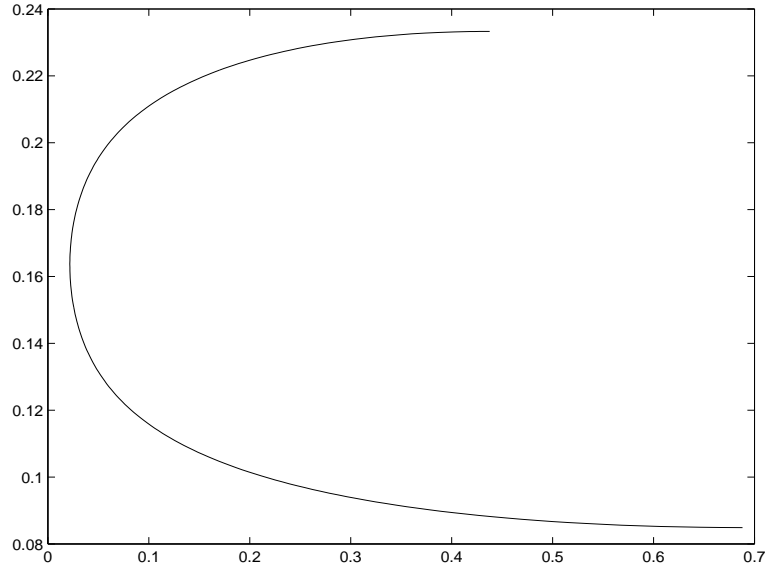


Figure 3: A typical outer surface with ψ at the top corresponding to $-\pi$.

initial condition to give ODE45. This is achieved by knowing the r and u where the surface is vertical. From a table in Hartland and Hartley [4] that has the height u of the vertical point associated with r for a range of values, we interpolate for u at a given r . Also we must rescale, as the table is for values of the capillary constant $\kappa = 1$, using $\frac{1}{\sqrt{\kappa}}u(\sqrt{\kappa}r) \rightarrow u(r)$. The constant κ is defined in the next section. This gives us the initial conditions to start with: $u_{-\frac{\pi}{2}}$, at $r_{-\frac{\pi}{2}}$ at, of course, $\psi = -\frac{\pi}{2}$. Now we just integrate from $\psi = -\frac{\pi}{2}$ to the angle we need, finding the radius r at which that happens. Now to complete

the surface we need only find the proper $u_{-\frac{\pi}{2}}$ and $r_{-\frac{\pi}{2}}$ that give the angle we need at \bar{r} , in a similar manner as we did for the inner surfaces. For this task we use FZERO again, to vary $r_{-\frac{\pi}{2}}$ and thereby varying $u_{-\frac{\pi}{2}}$, producing the initial conditions desired. However this surface is not typically the surface we seek. This one ends where we want to begin, at \bar{r} (see figure), and does not continue

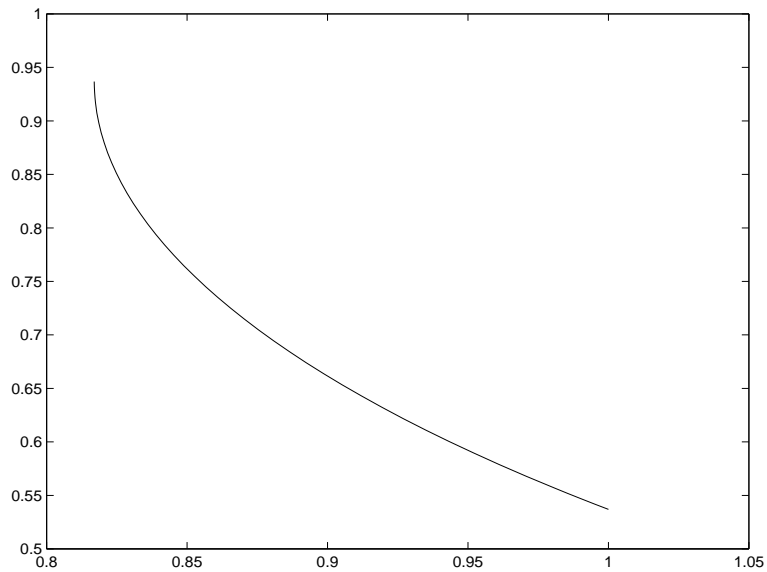


Figure 4: The sessile surface that we calculate, ending at \bar{r} .

out to $r = \infty$. We wait to deal with this till the end of the program, but the solution is to take the ending data from the FZERO function and use that as initial conditions and integrate out to a sufficiently large radius to illustrate the behavior. It is sufficient to use this ending value at \bar{r} to construct the solution to the larger problem at hand though, so we wait until that is done to get the proper surface, as it speeds the calculations to do so.

So we may look at these sessile surfaces as functions of their contact angle $\bar{\psi}$ at the radius \bar{r} , and letting the other boundary conditions be determined as above. In doing so we set the context for the larger problem.

3 The Free Boundary Problem

Now we consider the larger problem of matching all these surfaces with the proper angles at \bar{r} . There arises the capillary constant, which is related to the properties of the materials: $\kappa = \frac{\Delta\rho}{\sigma}g$. In our cases, we have three: $\kappa_{01} = \frac{\rho_1 - \rho_0}{\sigma_{01}}g$, $\kappa_{02} = \frac{\rho_2 - \rho_0}{\sigma_{02}}g$, and $\kappa_{12} = \frac{\rho_2 - \rho_1}{\sigma_{12}}g$. Define M to be the operator on the right hand side of (3).

Before we begin matching the surfaces we must talk about their configuration. The upper surface, u , is a sessile drop, the lower, v , an inverted sessile drop, or using the duality with capillary surfaces ([2] p38), a capillary surface, and the outer, w , is an exterior sessile surface. The angles between these surfaces are determined by the surface tensions, which are properties of the fluids involved. Figure 5 is a useful guide to defining these angles. We define the angle between u and v to be γ_{02} , the angle between v and w to be γ_{01} , and the angle between w and u to be γ_{12} , which is of course just $2\pi - \gamma_{01} - \gamma_{02}$ as we have $\gamma_{01} + \gamma_{02} + \gamma_{12} = 2\pi$. Also, as these σ quantities are in a force balance, you must have $0 \leq \gamma_{01}, \gamma_{02}, \gamma_{12} \leq \pi$. Now we find these angles explicitly from the law of cosines and this force balance of the surface tensions:

$$\gamma_{02} = \pi - \arccos\left(\frac{\sigma_{01}^2 + \sigma_{12}^2 - \sigma_{02}^2}{2\sigma_{01}\sigma_{12}}\right), \quad (7)$$

and

$$\gamma_{01} = \pi - \arccos\left(\frac{\sigma_{12}^2 + \sigma_{02}^2 - \sigma_{01}^2}{2\sigma_{12}\sigma_{02}}\right). \quad (8)$$

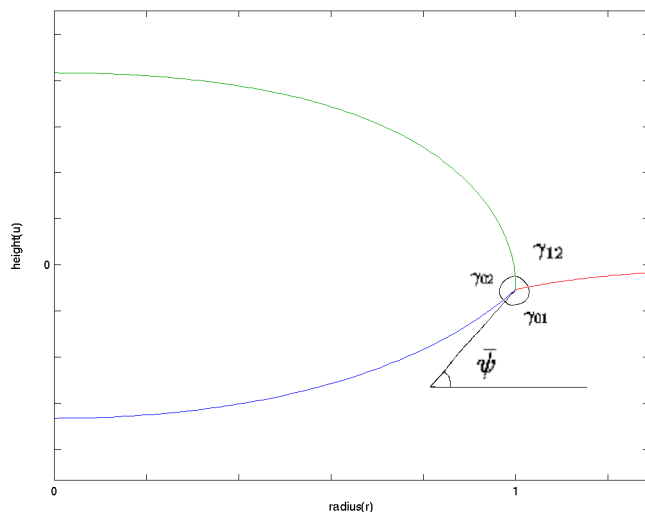


Figure 5: The angles between the surfaces.

Also if we define $\bar{\psi}$ here and in what follows to be the contact angle of the surface v at the radius \bar{r} , then we must have $0 < \bar{\psi} < \gamma_{02}$. If this were not the case then the drop would be 'tipped' up too far and the surface u would be inclined instead of beginning at a zero slope and decreasing. In fact if $\bar{\psi} = \gamma_{02}$, then the fact that the supplement of the sum of γ_{02} and the complement of $\bar{\psi}$ is equal to $\frac{\pi}{2} - \gamma_{02} + \bar{\psi}$, then we have $\frac{\pi}{2}$ for this angle and u is flat. Stated differently, the slope at \bar{r} is zero, and thus the surface u is flat, and any larger angle $\bar{\psi}$ should be excluded from our consideration. This also gives us our first relation between the different boundary conditions. If we have $\bar{\psi}$ for the surface v , then the proper contact angle for the surface u would be $\bar{\psi}_u := \gamma_{02} - \bar{\psi}$. This angle is down from horizontal due to inverting, a process we describe momentarily.

Now we need to describe the character of the outer surface, w . There are several possibilities available: w slopes down to 0, w slopes up to 0, and one case of each where w is curled over. We obtain these cases by looking at simple hand drawn pictures and exploring the possibilities considering different angle configurations. These last two cases may seem physically unlikely, but are included so as not to disallow them. We have two cases for $0 \leq \gamma_{02} \leq \frac{\pi}{2}$: if $\gamma_{01} \leq \gamma_{12}$ then w slopes down, and if not then w slopes up. However if we consider $\gamma_{02} > \frac{\pi}{2}$, the situation is much more complicated. We break this into two classes of situations: if $\bar{\psi} \leq \frac{\pi}{2}$ and the interface is below the middle of the drop, or else $\bar{\psi} > \frac{\pi}{2}$ with the interface above the middle of the drop. There are three configurations contained in each class. First we consider $\bar{\psi} \leq \frac{\pi}{2}$. If $0 \leq \bar{\psi} \leq (\frac{\pi}{2} - \gamma_{01})$ then w is folded over and sloping down. If $(\frac{\pi}{2} - \gamma_{01}) \leq \bar{\psi} \leq (\pi - \gamma_{01})$ then w is merely sloping down. Now if $(\pi - \gamma_{01}) \leq \bar{\psi} \leq \gamma_{02}$ then w is merely sloping up. For the second class, where $\bar{\psi} > \frac{\pi}{2}$ we go through the three cases. If $0 \leq \bar{\psi} \leq (\pi - \gamma_{01})$ then w is merely sloping down. If $(\pi - \gamma_{01}) \leq \bar{\psi} \leq (\frac{3\pi}{2} - \gamma_{01})$, then w merely slopes up. Lastly if $(\frac{3\pi}{2} - \gamma_{01}) \leq \bar{\psi} \leq \gamma_{02}$ then w is folded over, sloping up.

So now, keeping in mind that the contact angle for w is defined for negative values, we can find a relation for the initial angle, $\bar{\psi}_w$, of w that depends on the value for $\bar{\psi}$, as we did for $\bar{\psi}_u$. Now if w is sloping up we have $\bar{\psi}_w := \pi - \bar{\psi} - \gamma_{01}$ and the surface w is actually inverted from that described in Chapter 2, which

we achieve by replacing w by $-w$. On the other hand, if w is sloping down we have $\bar{\psi}_w := \bar{\psi} + \gamma_{01} - \pi$, with w in its usual sense.

For the top, u , and the bottom, v , we can matching these surfaces at \bar{r} by eliminating the Lagrange multiplier λ . Define U and V by $u = -U - \frac{\lambda}{\kappa_{01}\sigma_{01}}$ and $v = V + \frac{\lambda}{\kappa_{12}\sigma_{01}}$. Then $MU = \kappa_{01}U$ and $MV = \kappa_{12}V$. Now $u(\bar{r}) = v(\bar{r})$ implies that

$$\lambda = -[U(\bar{r}) + V(\bar{r})] \frac{\kappa_{01}\sigma_{01}\kappa_{12}\sigma_{12}}{\kappa_{01}\sigma_{01} + \kappa_{12}\sigma_{12}}. \quad (9)$$

But if we put this back into the equations that we defined U and V with we get

$$u(\bar{r}) = v(\bar{r}) = \frac{\kappa_{12}\sigma_{12}V(\bar{r}) - \kappa_{01}\sigma_{01}U(\bar{r})}{\kappa_{01}\sigma_{01} + \kappa_{12}\sigma_{12}}, \quad (10)$$

which is what we were after.

Because we know how to solve the sessile equations numerically, now all we need to do to get the upper surface, u , to match the lower surface, v , at \bar{r} is to add a constant to both U and V , which obey the neumann boundary conditions and thus end at the proper angles.

This gives the value of $u = v$ at \bar{r} . We again use FZERO, this time to determine when the difference between u and w at the prescribed radius \bar{r} is zero. We write the difference of u and w as

$$F(\bar{\psi}) = \frac{\kappa_{12}\sigma_{12}V(\bar{r}) - \kappa_{01}\sigma_{01}U(\bar{r})}{\kappa_{01}\sigma_{01} + \kappa_{12}\sigma_{12}} - w(\bar{r}) \quad (11)$$

and vary $\bar{\psi}$ which is the contact angle at \bar{r} of the surface v . Now, as we have described before all these surfaces and explicitly their contact angles at \bar{r} are

functions of $\bar{\psi}$. So if we vary the value of $\bar{\psi}$ then we find the root of the function F that is known to exist [1], and thus match the surfaces at \bar{r} , ie. $u(\bar{r}) = v(\bar{r}) = w(\bar{r})$ and we have completed our task.

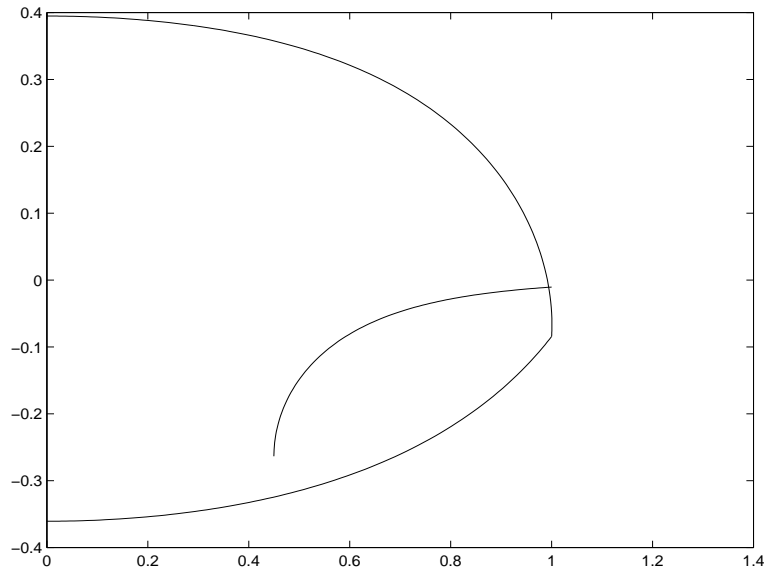


Figure 6: A typical iteration while varying $\bar{\psi}$.

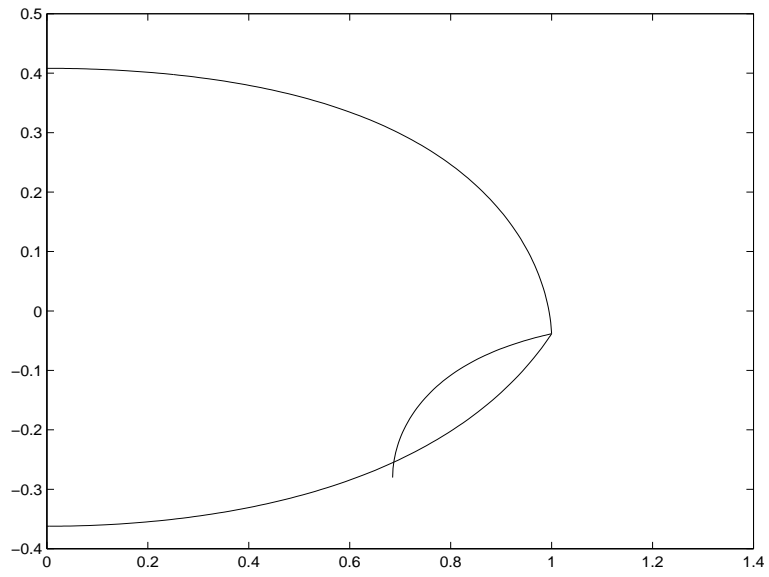


Figure 7: A converged case before we extend the outer surface.

4 Results

We have here a collection of pictures. The first picture is somewhat typical of the shape for most configurations. What follows is collection of more interesting cases. Some of these take some work to get. The main FZERO function

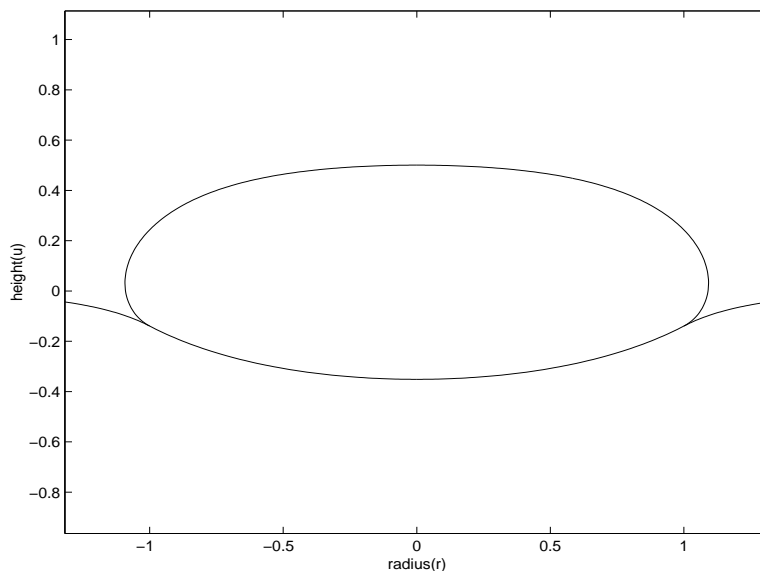


Figure 8: $\bar{r} = 1, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 10, \sigma_{02} = 20, \sigma_{12} = 30$.

that matches the outer surface to the inner surfaces at \bar{r} , sometimes encounters difficulties. This can be attributed to the built-in function not having the proper initial guess, or to difficulties in numerically determining a single capillary surface that is too close to an arc of a circle or too flat for small ψ . Some measures have been introduced to keep this to a minimum. Upon finding a case that does not converge, if we run the file `diff_surf.m`, it manually checks the interval $[0, \gamma_{02}]$ for a sign change in the function $F(\bar{\psi})$. In order to avoid

difficulty, it stays slightly away from the endpoints of this interval, as the differential equations cannot be numerically solved there. As already discussed, at one endpoint the upper surface will be flat. At the other endpoint the lower surface should be flat as well. This function returns a small interval around the sign change if it finds one and 0 otherwise. Then a small change in the driver function, changing the psibarguess variable to the output of diff_surf.m, gives a correct value unless u or v is indeed too close to a circular arc or too flat for ψ small.

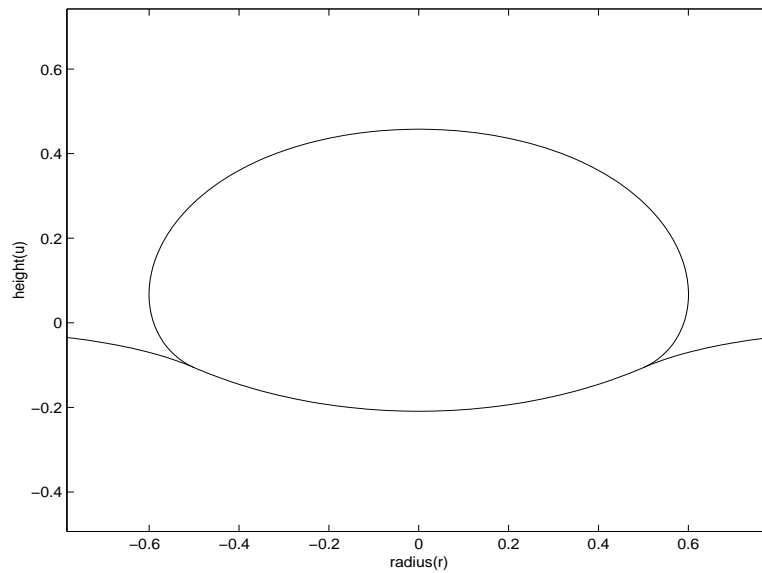


Figure 9: $\bar{r} = .5, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 10, \sigma_{02} = 20, \sigma_{12} = 30$.

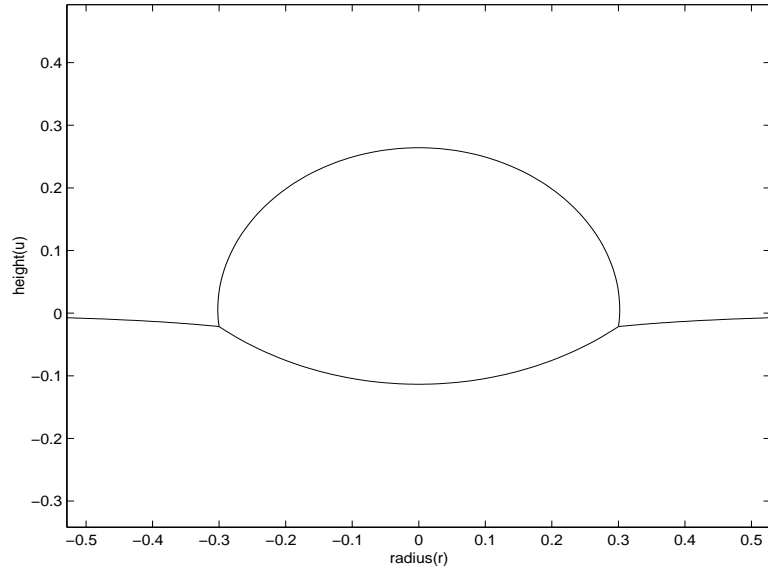


Figure 10: $\bar{r} = .3, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 10, \sigma_{02} = 15, \sigma_{12} = 20$.

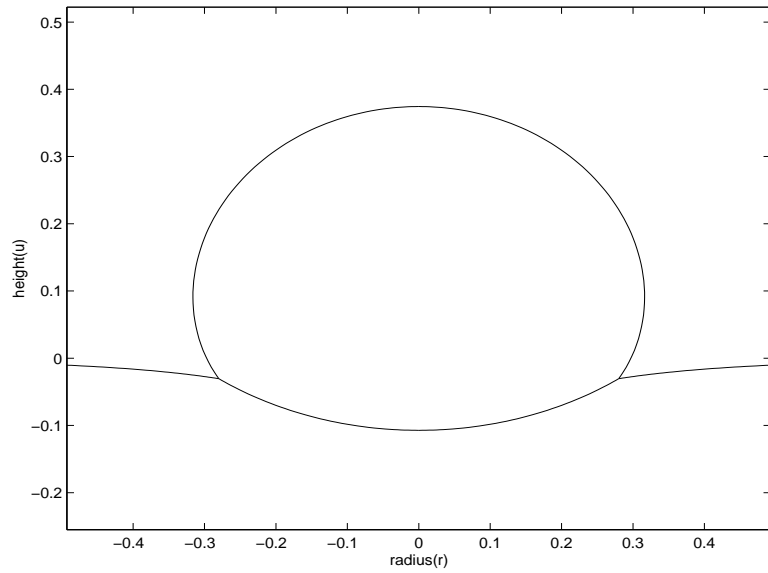


Figure 11: $\bar{r} = .28, \rho_0 = 0, \rho_1 = 7, \rho_2 = 20, \sigma_{01} = 12, \sigma_{02} = 13, \sigma_{12} = 23$.

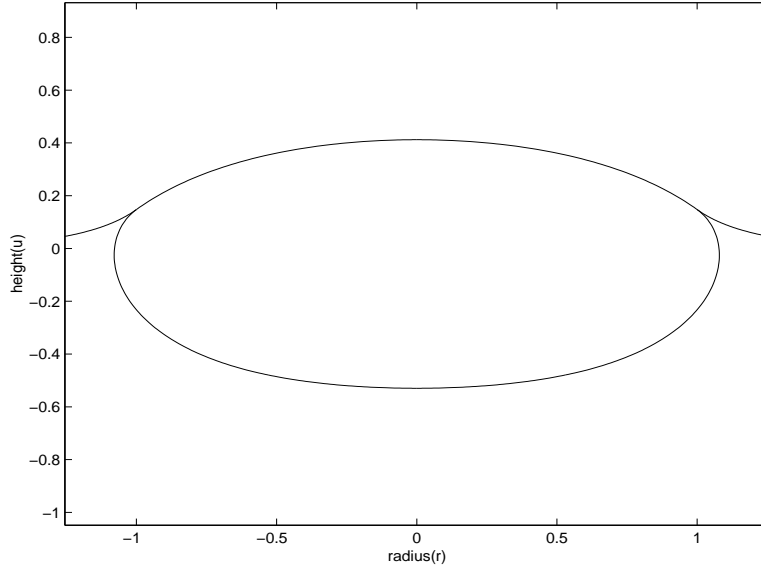


Figure 12: $\bar{r} = 1, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 25, \sigma_{02} = 13, \sigma_{12} = 12$.

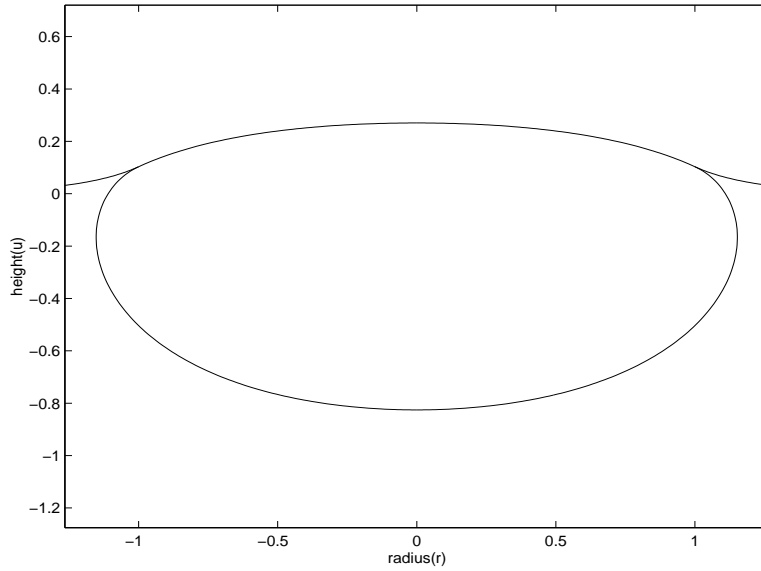


Figure 13: $\bar{r} = 1, \rho_0 = 0, \rho_1 = 15, \rho_2 = 20, \sigma_{01} = 25, \sigma_{02} = 13, \sigma_{12} = 12$.

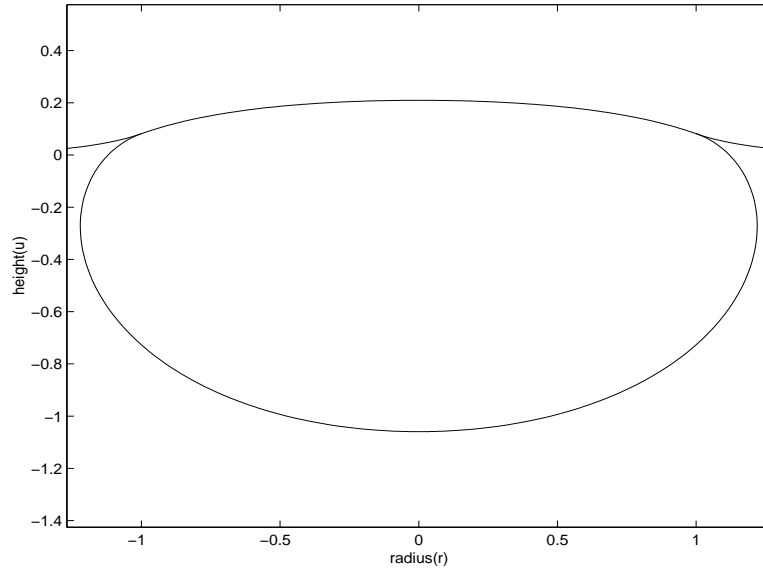


Figure 14: $\bar{r} = 1, \rho_0 = 0, \rho_1 = 17, \rho_2 = 20, \sigma_{01} = 25, \sigma_{02} = 13, \sigma_{12} = 12$.

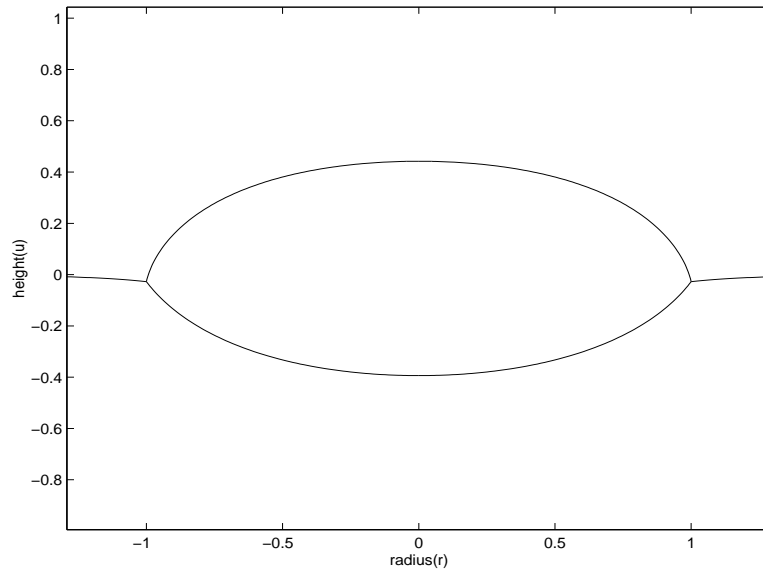


Figure 15: $\bar{r} = 1, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 15, \sigma_{02} = 15, \sigma_{12} = 20$.

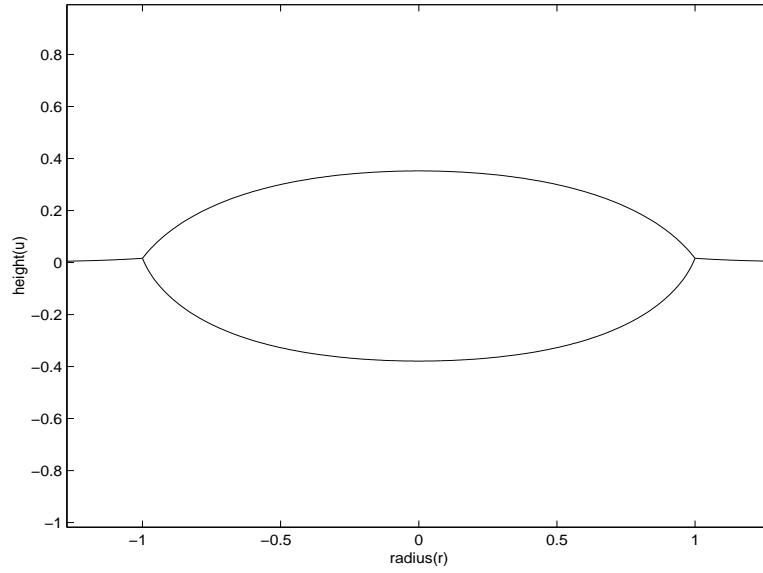


Figure 16: $\bar{r} = 1, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 15, \sigma_{02} = 13, \sigma_{12} = 12$.

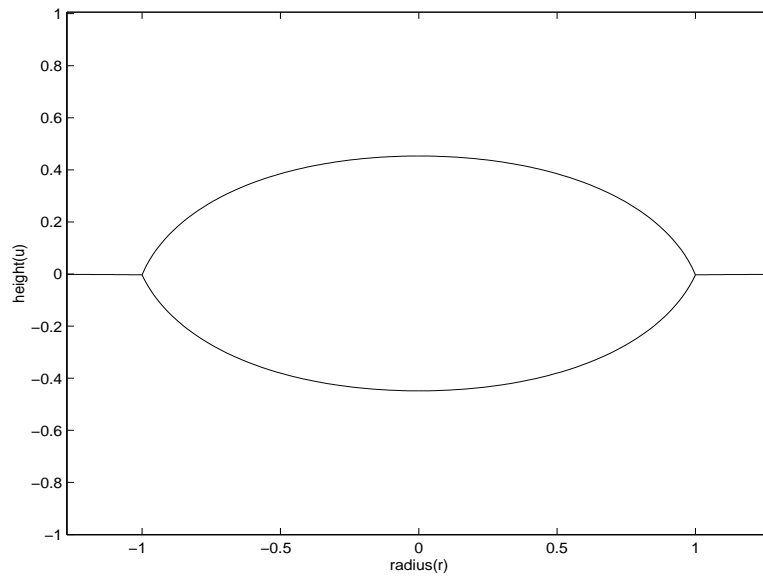


Figure 17: $\bar{r} = 1, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 19.5, \sigma_{02} = 15, \sigma_{12} = 20$.

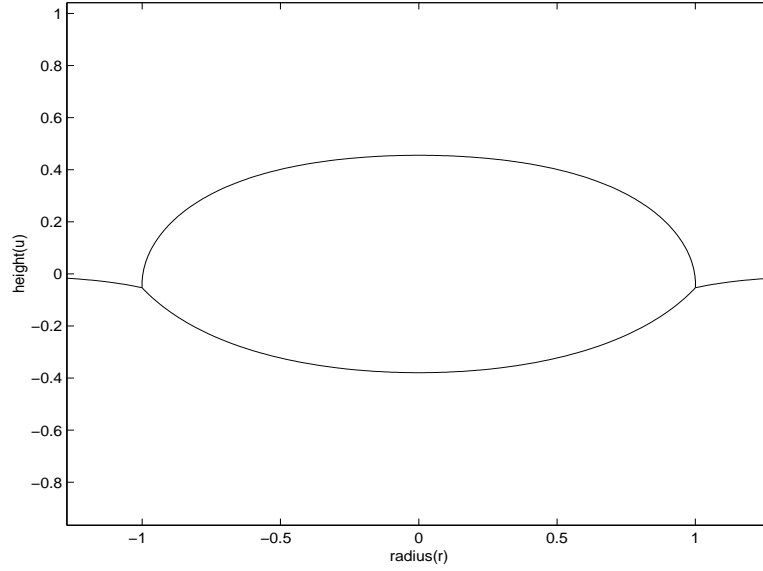


Figure 18: $\bar{r} = 1, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 12, \sigma_{02} = 13, \sigma_{12} = 20$.

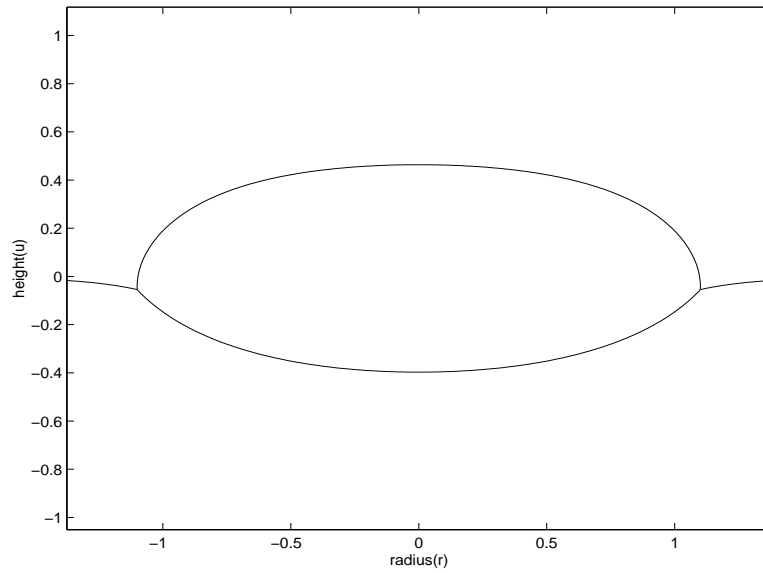


Figure 19: $\bar{r} = 1.1, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 12, \sigma_{02} = 13, \sigma_{12} = 20$.

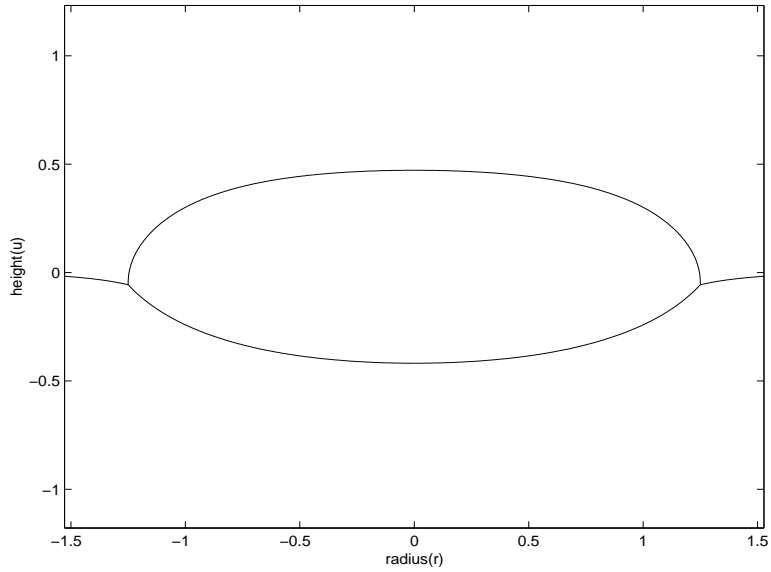


Figure 20: $\bar{r} = 1.25, \rho_0 = 0, \rho_1 = 10, \rho_2 = 20, \sigma_{01} = 12, \sigma_{02} = 13, \sigma_{12} = 20$.

Some conjectures arise from these pictures. First, if you decrease the radius, u and v approach circular arcs. Also, if you increase the radius, the drop flattens out and all the curvature is concentrated near \bar{r} . If any two of the surface tensions are equal, the outer surface is nearly flat.

Also, here is a table of $\bar{\psi}$ and volume for varying inputs of \bar{r} , ρ , and σ values. There are many parameters and we do not pick any particular one to sort by as they are all varied at some point or another. The volume of each drop has been added to the computations and it seems to conform to what one would expect.

Table 1: Data from various configurations of \bar{r} , ρ , and σ .

\bar{r}	ρ_0	ρ_1	ρ_2	σ_{01}	σ_{02}	σ_{12}	ψ	Volume
1.0	0	10	20	10	10	15	0.9108	1.5122
1.0	0	10	20	10	20	30	0.5061	2.2206
0.5	0	10	20	10	20	30	0.4290	0.5167
1.0	0	5	20	10	15	20	0.5773	2.0071
1.0	0	10	20	15	15	20	0.9525	1.5853
1.0	0	10	20	18	15	20	1.0913	1.6987
1.0	0	10	20	19	15	20	1.1384	1.7302
1.0	0	10	20	19.5	15	20	1.1622	1.7449
1.0	0	10	20	19.75	15	20	1.1743	1.7521
1.0	0	10	20	19.85	15	20	1.1792	1.7549
1.0	0	10	20	19.9	15	20	1.1815	1.7563
1.0	0	10	12	10	15	20	0.8782	0.3827
1.0	0	10	15	15	15	20	1.0432	1.5881
1.0	0	10	14	15	15	20	1.0679	1.6073
1.0	0	10	20	16	15	14	1.1282	1.4190
1.0	0	10	20	15	14	13	1.1340	1.3959
1.0	0	10	20	14	13	12	1.1408	1.3706
1.0	0	10	20	15	13	12	1.2138	1.4085
1.0	0	10	20	16	13	12	1.2887	1.4433
1.0	0	10	20	17	13	12	1.3661	1.4765
1.0	0	10	20	20	13	12	1.6212	1.5868
1.0	0	10	20	24	13	12	2.1129	1.9306
1.0	0	10	20	25	13	12	2.4888	2.3601
1.0	0	15	20	25	13	12	2.6889	3.2093
1.0	0	17	20	25	13	12	2.7839	4.2006
1.0	0	7	20	12	13	22	0.6947	2.0475
1.0	0	7	20	12	13	23	0.6631	2.1595
0.5	0	7	20	12	13	23	0.6193	0.4150
0.3	0	7	20	12	13	23	0.5601	0.1200
0.28	0	7	20	12	13	23	0.5514	0.1013
1.0	0	10	20	12	13	20	0.8463	1.5868
1.1	0	10	20	12	13	20	0.8501	2.0231
1.2	0	10	20	12	13	20	0.8529	2.5154
1.25	0	10	20	12	13	20	0.8541	2.7826
1.255	0	10	20	12	13	20	0.8542	2.8101

References

References

- [1] Elcrat, Alan, and David Siegel, personal communication.
- [2] Finn, Robert. *Equilibrium Capillary Surfaces*. New York: Springer-Verlag, 1986.
- [3] Gibbs, Stephen. *On Floating Drops and Various Methods of Measuring Surface Tension*. Thesis Research Proposal, University of Waterloo, 1989, unpublished.
- [4] Hartland, Stanley, and Richard W. Hartley. *Axisymmetric Fluid-Liquid Interfaces*. New York: Elsevier Scientific Publishing Company, 1976.
- [5] Hemphill, Kevin. *Bubble Development of Known Radius from Capillary surfaces*. Thesis, Wichita State University, 1999.
- [6] Vogel, Thomas I., *Symmetric Unbounded Liquid Bridges*, Pacific Journal of Mathematics, 103 (1982): 205-241.

Appendix

5 Appendix

Here are the m-files that solve this problem. The comments should help to clarify the usage of the program, however, there are quite a few functions. The first file is main.m, and it calls matchlight.m from the built-in function FZERO. From there there are two functions called. The first one is drop2.m and it calculates the inner surfaces, calling Usurf.m twice. This in turn calls uode.m and sessile.m. The second thing that matchlight calls is angleswitch.m, which determines the configuration of the outer surface, and it calls the Osurf.m function. This in turn calls interp.m, oode.m, and sessile.m. Lastly there is diff_surf.m. This is to manually search the interval between 0 and γ_{02} for the roots of the function F , which we need to get some configurations to converge to a solution. Then in the driver function the variable psiguess needs to be set to out, and run again.

```

% driver file for the drop problem
% adjust the densities and the surface tensions to
% find drops with those attributes.

% Declare the global variable for the initial guesses
% to be used to speed calculations and adapt the guess to one
% closer to the desired value.
global iguess
iguess=[.5;.5;.5]';

% set the radius
r=1;

% Densities:
rho0 = 0;
rho1 = 10;
rho2 = 20;

% Surface tensions:
sigma01 = 10;
sigma02 = 10;
sigma12 = 15;

% determine the kappa's
% where here acceleration due to gravity is g
g = 9.80665;
k01 = (rho1-rho0)*g/sigma01;
k02 = (rho2-rho0)*g/sigma02;
k12 = (rho2-rho1)*g/sigma12;

% determine the angles from the surface tensions
% using the law of cosines
gamma02 = pi-acos((sigma02^2-sigma01^2-...
    sigma12^2)/(-2*sigma01*sigma12));
gamma01 = pi-acos((sigma01^2-sigma12^2-...
    sigma02^2)/(-2*sigma12*sigma02));

% determine via fzero the correct psibar
% here tolerance is set to 10^-6
opt = optimset('tolx',1e-6);
psiguess = gamma02/3;
[psibar,fval,exitflag,output] = fzero('matchlight',psiguess...
    ,opt,k01,k02,k12,sigma01,sigma02,sigma12,...
    gamma01,gamma02,r);

```

```

% if fzero did not find a psibar that is a root of matchlight
% then we display an error message
if exitflag < 0
    error('failed to find a psibar that made the drop ...
        and the outer surface meet.')
end

% create the final surfaces.
% beginning with the upper and lower surfaces
[v u] = drop2(gamma02,psibar,k01,k12,sigma01,sigma12,r);

% picking which outer surface to plot for the graph
% this is just to generate the outer surface from rbar out
% to a reasonable radius
[psi w slope] = angleswitch(gamma02,gamma01,psibar,k02,r);
[m,n] = size(w);
[p,q] = size(psi);

switch slope
    case 'down'
        angle = [psi(p) .3*psi(p)];
        options = odeset('AbsTol',1e-8,'RelTol',1e-8);
        [theta,w] = ode45('sessile',angle,...
            [w(m,1) w(m,2)],options,k02);

    case 'up'
        angle = [psi(p) .3*psi(p)];
        options = odeset('AbsTol',1e-8,'RelTol',1e-8);
        [theta,w] = ode45('sessile',angle,...
            [w(m,1) -w(m,2)],options,k02);
        w(:,2) = -w(:,2);

    case 'fold down'
        angle = [psi(p) .3*psi(p)];
        options = odeset('AbsTol',1e-8,'RelTol',1e-8);
        [theta,w] = ode45('sessile',angle,...
            [w(m,1) w(m,2)],options,k02);

    case 'fold up'
        angle = [psi(p) .3*psi(p)];
        options = odeset('AbsTol',1e-8,'RelTol',1e-8);
        [theta,w] = ode45('sessile',angle,...
            [w(m,1) -w(m,2)],options,k02);
        w(:,2) = -w(:,2);

end

% plotting the surface
% the pairs that begin with the negative value correspond
% to the values on the left half plane
plot(v(:,1),v(:,2),-v(:,1),v(:,2),u(:,1),u(:,2),-u(:,1),...
    u(:,2), w(:,1),w(:,2),-w(:,1),w(:,2))
axis equal
xlabel('radius(r)')
ylabel('height(u)')

```

```

function error=matchlight(psiguess,k01,k02,k12,sigma01,...
    sigma02,sigma12,gamma01,gamma02,r)
% this function computes the error between the drop and the
% outer surface for the light surface.

% call the light drop function to calculate the upper
% and lower surfaces for the interface point.
[v u] = drop2(gamma02,psiguess,k01,k12,sigma01,sigma12,r);

% temporary plotting used for error checking and debugging
% figure(9)
% plot(u(:,1),u(:,2),v(:,1),v(:,2))

% call the outer surface
[theta w slope] = angleswitch(gamma02,gamma01,psiguess,k02,r);

% temporary plotting used for error checking and debugging
% figure(10)
% plot(u(:,1),u(:,2),v(:,1),v(:,2),w(:,1),w(:,2))
% pause

% getting the proper elements from each
% of the surface height vectors
[m1 n1] = size(u);
[m2 n2] = size(w);

heightu = u(m1,n1);
heightw = w(m2,n2);

% determining the discrepancy between the inner surfaces at rbar
% and the outer surface there.
error = heightu-heightw;

```

```

function [v,u]=drop2(gamma02,psibar,k01,k12,sigma01,sigma12,r)
% This function constructs the light drop
% with a upper sessile surface and a lower sessile surface
%
% Input:
% gamma02      angle between the two surfaces
% psibar       angle of the lower surface from horizontal
% k01          capillary constant for the upper surface
% k12          capillary constant for the lower surface
% sigma01     surface tension for the upper surface
% sigma12     surface tension for the lower surface
% r           radius of the interface point
%
% Output:
% v           radius and height matrix for the lower surface
% u           radius and height matrix for the upper surface

% declare global variable for the initial conditions
global iguess

% resetting these if they get too close to 0, as that is not a
% good value to start with
if iguess(1) < .05
    iguess(1) = .5;
end
if iguess(2) < .05
    iguess(2) = .5;
end

% call the sessile surface function
% giving it the cap. constant, radius, and angle
[psi v] = Usurf(k12,r,psibar,iguess(2));

% resetting the initial value guess for the lower surface
% to speed calculation and help find good values
iguess(2) = v(1,2);

% call the upper surface function
% giving it the cap. constant, radius, and angle
% here the angle is gamma02-psibar as the surface
% is inverted
[psi u] = Usurf(k01,r,gamma02-psibar,iguess(1));

% resetting the initial value guess for the upper surface
% to speed calculation and help find good values
iguess(1) = u(1,2);

% matching the heights of the two surfaces
% calculating lambda to adjust the height of both surfaces
% so that they match
% this formula came from the waterloo fax
[m1 n] = size(v);
[m2 n] = size(u);
lambda = -[u(m2,2)+v(m1,2)]*k01*sigma01*k12*...
    sigma12/(k01*sigma01+k12*sigma12);

v(:,2) = v(:,2)+lambda/(k12*sigma12);

u(:,2) = -u(:,2)-lambda/(k01*sigma01);

```

```

function [psi,u]= Usurf(k,r,psibar,u0guess)
% This function first determines the correct initial
% value for u0 using fzero, then constructs the sessile surface
%
% Input:
% k          capillary constant
% r          radius
% psibar     ending angle from horizontal
% u0guess    the initial guess for the fzero function
%
% Output:
% psi        The vector of angles at which there are values
% u          The matrix of radii and height for the psi values
% u0         return the initial value from the fzero function
%            used for error checking

% find the correct initial value for u0
% setting the options to 10^-6
opt = optimset('tolx',1e-6);
[u0,fval,exitflag,output] = fzero('uode',u0guess,opt,k,r,psibar);

% error message if fzero fails to find the correct value for u0
if exitflag < 0
    error('failed to find an initial guess that made the ...
          sessile surface meet the needed values at rbar.')
end

% now compute the correct surface with options set to 10^-8
options = odeset('AbsTol',1e-8,'RelTol',1e-8);
[psi,u] = ode45('sessile',[0 psibar],[0 u0],options,k);

% error checking, used in debugging and testing.
% [m n] = size(u);
% end_error = r-u(m,1);

```

```

function error=matchlight(psiguess,k01,k02,k12,sigma01,...
    sigma02,sigma12,gamma01,gamma02,r)
% this function computes the error between the drop and the
% outer surface for the light surface.

% call the light drop function to calculate the upper
% and lower surfaces for the interface point.
[v u] = drop2(gamma02,psiguess,k01,k12,sigma01,sigma12,r);

% temporary plotting used for error checking and debugging
% figure(9)
% plot(u(:,1),u(:,2),v(:,1),v(:,2))

% call the outer surface
[theta w slope] = angleswitch(gamma02,gamma01,psiguess,k02,r);

% temporary plotting used for error checking and debugging
% figure(10)
% plot(u(:,1),u(:,2),v(:,1),v(:,2),w(:,1),w(:,2))
% pause

% getting the proper elements from each
% of the surface height vectors
[m1 n1] = size(u);
[m2 n2] = size(w);

heightu = u(m1,n1);
heightw = w(m2,n2);

% determining the discrepancy between the inner surfaces at rbar
% and the outer surface there.
error = heightu-heightw;

```

```

function error=uode(u0guess,k,r,psibar)
% This function is called by fzero to determine the proper
% initial condition to meet the correct angle at
% the prescribed radius
%
% Input:
% u0guess      initial guess for the initial condition
% k            capillary constant
% r           radius
% psibar      ending angle from horizontal
%
% Output:
% error       the difference between the final radius we want
%            and the radius that a u0 gives

% setting tolerance to 10^-8
options = odeset('AbsTol',1e-8,'RelTol',1e-8);

% calling the ode solver, ode45, integrating from
% psi=0 to psi=psibar.
[psi u] = ode45('sessile',[0 psibar],[0 u0guess],options,k);

% returning the difference between r and the radius
[m n] = size(u);
error = r-u(m,1);

% error checking. this was used in debugging and to get
% ideas of the behavior of the program
% u0guess;
% rbar = u(m,1);
% psibar_error = psibar-psi(m,1);
% if error == 1
%     u(:,1);
% end

```

```

function out1=sessile(psi,u,flag,k)
% This function is the ODEfile for the different surfaces
%
% Input:
% psi          angle that is integrated to
% u            initial conditions of the diff. eq.: [radius height]'
% flag        Currently set this equal to [] for a placemaker
% k            capillary constant
%
% Output:
% out1         returns the values of the solution matrix

% the first condition is l'Hopital's rule for the second
% condition this is done because the DE's are
% undefined at r,psi = 0.
if psi == 0
    out1 = [2/(k*u(2)) 0]';
else
    out1 = [u(1)*cos(psi)/((k*u(1)*u(2)-sin(psi)))
            u(1)*sin(psi)/(k*u(1)*u(2)-sin(psi))];
end

```

```

function [theta,w,slope]=angleswitch(gamma02,gamma01,...
    psibar,k,r)
% to determine which version on the outer surface to use
% and pass the needed information along to Osurf

% determine the angle gamma12 from the other two angles
gamma12 = 2*pi-gamma01-gamma02;

% this if condition finds the proper configuration of the
% outer surface and then send the necessary information
% to Osurf.
if gamma02 <= pi/2
    if gamma01 <= gamma12
        % slopes down here
        slope = 'down';
        [theta w] = Osurf(k,r,psibar+gamma01-pi);

    elseif gamma01 > gamma12
        % put slopes up here
        slope = 'up';
        [theta w] = Osurf(k,r,pi-psibar-gamma01);
        w(:,2) = -w(:,2);
    end

elseif gamma02 > pi/2
    if psibar <= pi/2-gamma01
        % outer surface is folded over sloping down
        slope = 'fold down';
        [theta w] = Osurf(k,r,psibar+gamma01-pi);

    elseif (pi/2-gamma01 < psibar)&...
        (psibar < pi-gamma01)&(psibar <= pi/2)
        % outer surface slopes down
        slope = 'down';
        [theta w] = Osurf(k,r,psibar+gamma01-pi);

    elseif (psibar >= pi-gamma01)&(psibar <= pi/2)
        % outer surface slopes up
        slope = 'up';
        [theta w] = Osurf(k,r,pi-psibar-gamma01);
        w(:,2) = -w(:,2);

    elseif (psibar <= pi-gamma01)&(psibar > pi/2)
        % outer surface slopes down
        slope = 'down';
        [theta w] = Osurf(k,r,psibar+gamma01-pi);

```

```
elseif (pi-gamma01 < psibar)&...
    (psibar < 3*pi/2-gamma01)&(psibar > pi/2)
    % outer surface slopes up
    slope = 'up';
    [theta w] = Osurf(k,r,pi-psibar-gamma01);
    w(:,2) = -w(:,2);

elseif (psibar >= 3*pi/2-gamma01)
    % outer surface is folded over sloping up
    slope = 'fold up';
    [theta w] = Osurf(k,r,pi-psibar-gamma01);
    w(:,2) = -w(:,2);

end

end
```

```

function [theta,w]= Osurf(k,r,thetabar)
% This function first determines the correct initial
% value for u0 using fzero, then constructs the outer surface
%
% Input:
% k          capillary constant
% r          radius
% thetabar  ending angle from horizontal
%           NOTE: this is opposite from the rest of the psibar's
%
% Output:
% theta      The vector of angles at which there are values
% u          The matrix of radii and height for the psi values

% Find the correct initial values for r and w
% using the guess for the radius close to the radius desired
% setting the tolerance of fzero to 10^-6
% where here the important output being the value for
% the radius at the vertical point
rguess = r-.1;
opt = optimset('tolx',1e-6);
[rpi2,fval,exitflag,output] = fzero('oode',rguess,opt,...
    k,r,thetabar);

% error message if the fzero call does not
% produce the needed value.
if exitflag < 0
    error('failed to find an initial guess that made the...
        outer surface match the values needed at rbar.')
end

% now compute the correct surface with options set to a
% tolerance of 10^-8. The rescaling from kappa = 1 is
% necessary here to account for the fact that the table
% from hartland and hartley has kappa = 1
options = odeset('AbsTol',1e-8,'RelTol',1e-8);
[theta,w] = ode45('sessile',[-pi/2 thetabar],...
    [rpi2 (1/sqrt(k))*interp(sqrt(k)*rpi2)],options,k);

```

```

function error=matchlight(psiguess,k01,k02,k12,sigma01,...
    sigma02,sigma12,gamma01,gamma02,r)
% this function computes the error between the drop and the
% outer surface for the light surface.

% call the light drop function to calculate the upper
% and lower surfaces for the interface point.
[v u] = drop2(gamma02,psiguess,k01,k12,sigma01,sigma12,r);

% temporary plotting used for error checking and debugging
% figure(9)
% plot(u(:,1),u(:,2),v(:,1),v(:,2))

% call the outer surface
[theta w slope] = angleswitch(gamma02,gamma01,psiguess,k02,r);

% temporary plotting used for error checking and debugging
% figure(10)
% plot(u(:,1),u(:,2),v(:,1),v(:,2),w(:,1),w(:,2))
% pause

% getting the proper elements from each
% of the surface height vectors
[m1 n1] = size(u);
[m2 n2] = size(w);

heightu = u(m1,n1);
heightw = w(m2,n2);

% determining the discrepancy between the inner surfaces at rbar
% and the outer surface there.
error = heightu-heightw;

```

```

function error=oode(rguess,k,rbar,thetabar)
% This function is called by fzero to determine the proper
% initial condition to meet the correct angle at the
% prescribed radius
%
% Input:
% k          capillary constant
% rguess     guess radius
% thetabar   ending angle from horizontal
% rbar       ending radius
%
% Output:
% error      the difference between the final radius we want
%            and the radius that a w0 gives

% setting tolerance to 10^-8
options = odeset('AbsTol',1e-8,'RelTol',1e-8);

% calling the ode solver, ode45, integrating from
% theta=-pi/2, the vertical point to theta=thetabar,
% the desired angle at rbar
% here we need to rescale to account for kappa = 1
% in the table from hartland and harley, to a genral kappa
[theta w] = ode45('sessile',[-pi/2 thetabar],...
    [rguess (1/sqrt(k))*interp(sqrt(k)*rguess)],options,k);

% returning the error
[m n] = size(w);
error = rbar-w(m,1);

```

```

function wpi2 = interp(r)
% This function finds a height that corresponds with
% the radius, using the interpolation of the 72 points below
% rtheta = values of rext at theta=pi/2
% wtheta = values of uext at theta=pi/2

rtheta=[1.66139E-4 1.96171E-4 2.30101E-4 2.68422E-4 3.11686E-4 ...
        3.60506E-4 4.15569E-4 4.77640E-4 5.47575E-4 6.26330E-4 ...
        7.14970E-4 8.14686E-4 9.26810E-4 1.05282E-3 1.19438E-3 ...
        1.35334E-3 1.53176E-3 1.73194E-3 1.95645E-3 2.20815E-3 ...
        2.49023E-3 2.80626E-3 3.16021E-3 3.55649E-3 4.00004E-3 ...
        4.49636E-3 5.05155E-3 5.67245E-3 6.36664E-3 7.14259E-3 ...
        8.97844E-3 1.12688E-2 1.41237E-2 1.76789E-2 2.21024E-2 ...
        2.47026E-2 3.08308E-2 3.84363E-2 4.28978E-2 5.33872E-2 ...
        7.39397E-2 1.02039E-1 1.40144E-1 1.91188E-1 2.84893E-1 ...
        3.78175E-1 5.36685E-1 7.34386E-1 9.67774E-1 1.23124 ...
        1.51893 1.82571 2.14750 2.65159 2.99862 ...
        3.53160 3.89350 4.44408 4.81536 5.37739 ...
        5.75495 6.32485 6.70682 7.28234 7.66752 ...
        8.24722 8.63482 9.21769 9.60715 1.01925E1 ...
        1.05834E1 1.11708E1];

utheta=[1.58027E-3 1.83333E-3 2.11372E-3 2.42440E-3 2.76858E-3 ...
        3.14977E-3 3.57179E-3 4.03880E-3 4.55533E-3 5.12633E-3 ...
        5.75719E-3 6.45377E-3 7.22248E-3 8.07027E-3 9.00471E-3 ...
        1.00340E-2 1.11672E-2 1.24138E-2 1.37845E-2 1.52907E-2 ...
        1.69447E-2 1.87598E-2 2.07505E-2 2.29324E-2 2.53223E-2 ...
        2.79383E-2 3.07999E-2 3.39280E-2 3.73450E-2 4.10751E-2 ...
        4.95786E-2 5.96653E-2 7.15912E-2 8.56423E-2 1.02134E-1 ...
        1.11402E-1 1.32203E-1 1.56338E-1 1.69773E-1 1.99604E-1 ...
        2.52363E-1 3.15435E-1 3.88924E-1 4.71780E-1 5.92178E-1 ...
        6.84499E-1 8.01379E-1 9.03515E-1 9.87540E-1 1.05439 ...
        1.10694 1.14832 1.18121 1.21901 1.23861 ...
        1.26205 1.27470 1.29035 1.29906 1.31013 ...
        1.31645 1.32465 1.32941 1.33571 1.33942 ...
        1.34439 1.34736 1.35138 1.35381 1.35713 ...
        1.35914 1.36192];

%Interpolation done with a cubic spline
wpi2 = spline(rtheta,utheta,r);

```

```

function out = diff_surf(k01,k02,k12,sigma01,sigma02,sigma12,
gamma01,gamma02,r)
% this function checks the interval from 0 to gamma02
% looking for sign changes in the matching function
% if out = 0 there was no sign change, otherwise
% it returns an interval around where the sign change
% occurs.
% the main use is if the automated code fails to find the
% drop then this can be used to manually find the interval
% where the sign changes in F and then the output can be
% substituted for the initial guess in MAIN.m, where it will
% find the correct surface as long as this
% function does not return 0.

% finding a range of values for the angle psi, and restricting
% this away from the endpoints of possible solutions, as the
% differential equation solvers have problems there
psibar = linspace(0,gamma02,16);
psibar = psibar(4:13);

% Looping over the different angles,
% getting values of this function F
% that we want to find the zero of.
for i = 1:10
    diff(i) = matchlight(psibar(i),k01,k02,k12,sigma01,...
        sigma02,sigma12,gamma01,gamma02,r);
end

% looking for the sign change, and if there is none,
% the default for the output is zero.
out = 0;
for i = 1:9
    if sign(diff(i))+sign(diff(i+1)) == 0
        out = [psibar(i) psibar(i+1)];
    end
end
end

```