Towards an Analytical 2D Foam Solver: Solving Mancini's Equations and Related Problems

Bryan Gin–ge Chen
University of Pennsylvania
Department of Physics and Astronomy
chb@sas.upenn.edu

In this note I describe steps towards a 2D foam program which takes as input the topology of a 2D graph representing the edges and vertices of a bubble cluster, and draws images of an equilibrium foam with that topology and given pressures in the faces, in principle with exact solutions. I will explain how to (partially) solve Mancini's equations for 2D foam geometry in Mathematica 6.0 as well as explain other problems which arise in attempting to create such a program. A sufficient reference for this note is Mancini's 2005 thesis[1], in which Mancini's equations were first derived and stated (see in particular sections 2.4 and 2.5).

I will briefly explain here the structure of the notebook `20081222mancini.nb` so that the intrepid reader can immediately start playing with the interactive modules. First, evaluate the first two cells, which give the topological data for the bubbles in the notebook. Then run the third cell which contains all the functions which are used in the modules. My notebook `20081222mancini.nb` relies on Mathematica 6.0 for its interactivity. I belive that these functions and data ought to be compatible with earlier versions of Mathematica. The next few cells of the notebook are interactive `Manipulate` modules. The first example is the double bubble, where the two sliders control the pressures inside the two interior cells. The next is the triple bubble. Aside from the three sliders, there is also a checkbox "centers" which plots $\mathbf{O}, \mathbf{n}$ (blue dots and lines) (explained in the next section or in [1]) and the centers of the edges (red dots). Finally, there's also a checkbox "list" which displays the images in list form, rather than grid form. The next few modules work out several other topologies of bubbles, and figs 7 - 10 show what the typical output should be. The final section of the notebook consists of some matrices which will be explained in a later section of this document.

In the modules for the triple bubble and higher complexity bubbles, there are several images which change as you drag the sliders. The list of images above the horizontal line and the list below the line are in some sense the same, in that the edges in corresponding places arise from the same circles, but with different choice of "arcs" on the circles. I regard this as a limitation of the program and this will all be explained in this note.

1 Preliminaries

A 2D foam in equilibrium consists of a spatial graph of edges and vertices around faces, where each of the faces (including the exterior "face") is assigned a real number, the pressure. It is a consequence of equilibrium under length minimization that the edges of a foam will consist of arcs of circles, whose curvature is precisely the pressure difference across that edge divided by surface tension, and at a vertex, there will always be three edges meeting at precisely 120 degrees. Furthermore, the curvatures of the edges meeting at a vertex must sum to zero to

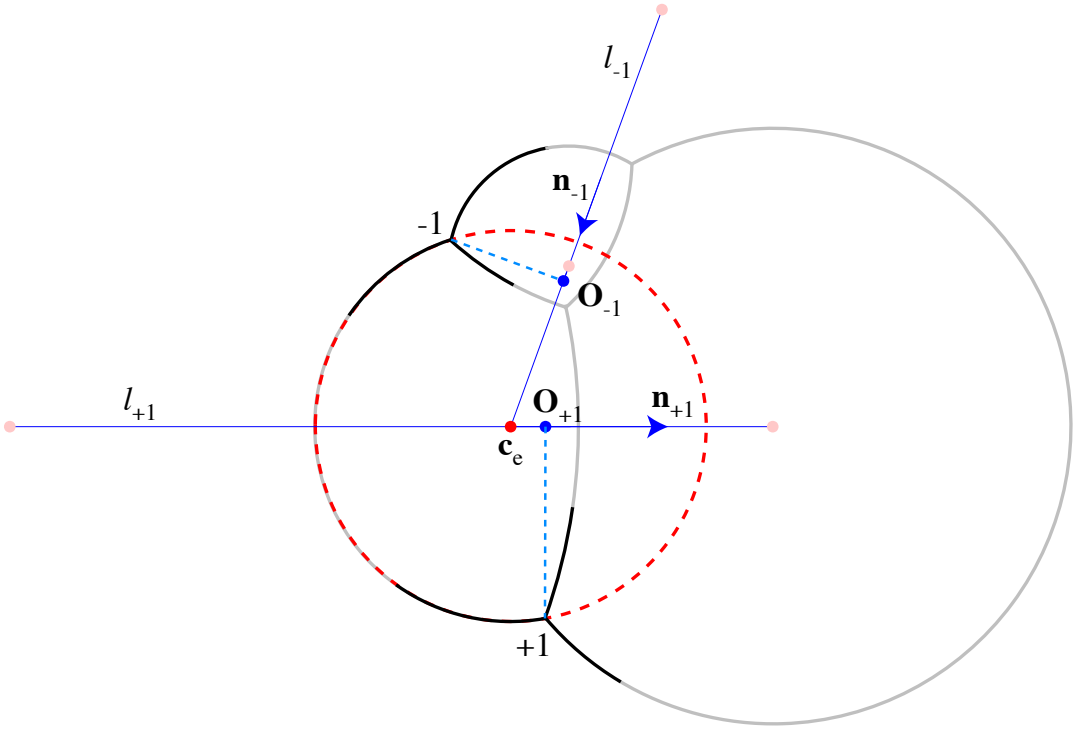

Figure 1: A triple bubble system with $\mathbf{O}_v$ (blue dots), $\mathbf{n}_v$ (blue arrows) and l_v (blue lines) marked for a pair of vertices. The labels for the two vertices here are their parities (see later section). The red / pink dots are the centers of the edges passing through the vertices – note that three such dots lie on each l_v.

ensure that faces of the foam may be assigned consistent pressures (up to a global constant). In this note we'll consider only finite bubble clusters – these will always have one "exterior" noncompact face.

If we are given the topology of a bubble cluster, for instance, a list of the E edges and V vertices along with the 2, respectively 3 faces out of $F+1$ (including the "exterior" face as one) that surround each, then the above conditions allow us to find the radii of curvatures of all the edges once we are given the F pressures in the interior faces. It's my goal in this note to explain how it is possible (in some cases) to compute the soap bubble geometry, namely, a computer generated image of the edges, given this initial data; see e.g. figs. 7 - 10.

Mancini's equations allow us to find the centers of the circles which define the edges. First, it's a consequence of the equilibrium conditions that if three edges meet in a vertex v, the centers of the three circles that the edges lie on must lie on a single straight line l_v. Hence, to each vertex v we may associate the point $\mathbf{O}_v$ which is the orthogonal projection of the position of the vertex onto the line associated with v and the unit vector $\mathbf{n}_v$, which is the unit vector along the line which points in a from the face with the greatest pressure to the face with the least pressure. See fig. 1 for an illustration of a few of the lines l and $\mathbf{O}, \mathbf{n}$.

Mancini's equations consist of one (2D) vector equation for every edge e:

$$\mathbf{O}_{v_1} + C'_{e,v_1}\mathbf{n}_{v_1} = \mathbf{O}_{v_2} + C'_{e,v_2}\mathbf{n}_{v_2}$$

where v_1 and v_2 are the indices of the vertices connected by edge e, and C'_{e,v_1} and C'_{e,v_2} are functions of the three pressures around v_j. If the pressures around v_j are $p_3 < p_2 < p_1$, and the pressures around e are p_2 and p_3 then (note that in this paper we will only consider cases where all pressures are distinct – the case of two equal pressures and three equal pressures (which are discussed in [1], but not implemented in my notebook) presumably will shed light on some of the issues we bring up later):

$$C'_{e,v_j} = \frac{1}{p_3 - p_2} \frac{p_2 + p_3 - 2p_1}{2\sqrt{m}}$$
$$m = \sum_k p_k^2 - \sum_{k<l} p_k p_l$$

We will call this case orientation 1 on an edge-vertex pair. If the pressures around e are p_1 and p_3 (orientation 2) then:

$$C'_{e,v_j} = \frac{1}{p_1 - p_3} \frac{p_1 + p_3 - 2p_2}{2\sqrt{m}}$$

In the final case, the pressures around e are p_1 and p_2 (orientation 3) and then:

$$C'_{e,v_j} = \frac{1}{p_2 - p_1} \frac{p_1 + p_2 - 2p_3}{2\sqrt{m}}$$

Each edge will have two orientations assigned to it, one to each vertex it connects. The labelling of the orientations (which turns out to be convenient for other reasons later) here is chosen because in the case that (e, v) has orientation 1, out of the three pressures p_1, p_2, p_3 in the faces around v_j, edge e separates p_2, p_3 and not p_1, in other words, in orientation j, p_j is the excluded pressure. Note that this also implies that if we consider the three edges e_l that go into a given vertex v, the three orientations of the pairs (e_l, v) will be 1,2,3. See fig. 2 for the orientations on a labeled triple bubble. The usefulness of the orientations will become clear in later sections.

Once this set of E 2D vector equations (and hence $2E$ scalar equations) is solved for the $3V = 2E$ unknowns, the center of the edge e is given by $\mathbf{O}_v + C'_{e,v}\mathbf{n}$ for either choice of v. Note that the system of equations is rotationally and translationally invariant, so we may choose some $\mathbf{O}$ to be (0,0) and some $\mathbf{n}$ to be (1,0) before we try to solve the entire system.

The original exposition of these equations may be found in Mancini's thesis[1]. I will emphasize some further features of Mancini's equations in this paragraph which are not found in Mancini's thesis. First, if we ignore the fact that $\mathbf{n}$ must be a unit vector, the constraints on the x and y components of any $\mathbf{O}_v$ and $\mathbf{n}_v$ are the same, and we have a set of *linear* equations in $\mathbf{O}$ and $\mathbf{n}$. Before we apply the unit vector constraint on $\mathbf{n}$, we have $2E$ linear equations in $4V$ unknowns.

After we find the centers of the circles defining the edges, we are still not finished, as we must find both the two angles on each of these circles defining the positions of the vertices, as well as "which side" of the vertices the edge lies. To my knowledge, algorithms for computing this data have not appeared yet in the literature before this note. As the method used in my code is somewhat involved (and this is where the orientation on the edges defined earlier will be used), it will be described in a section on its own after a section on my methods for solving Mancini's equations.

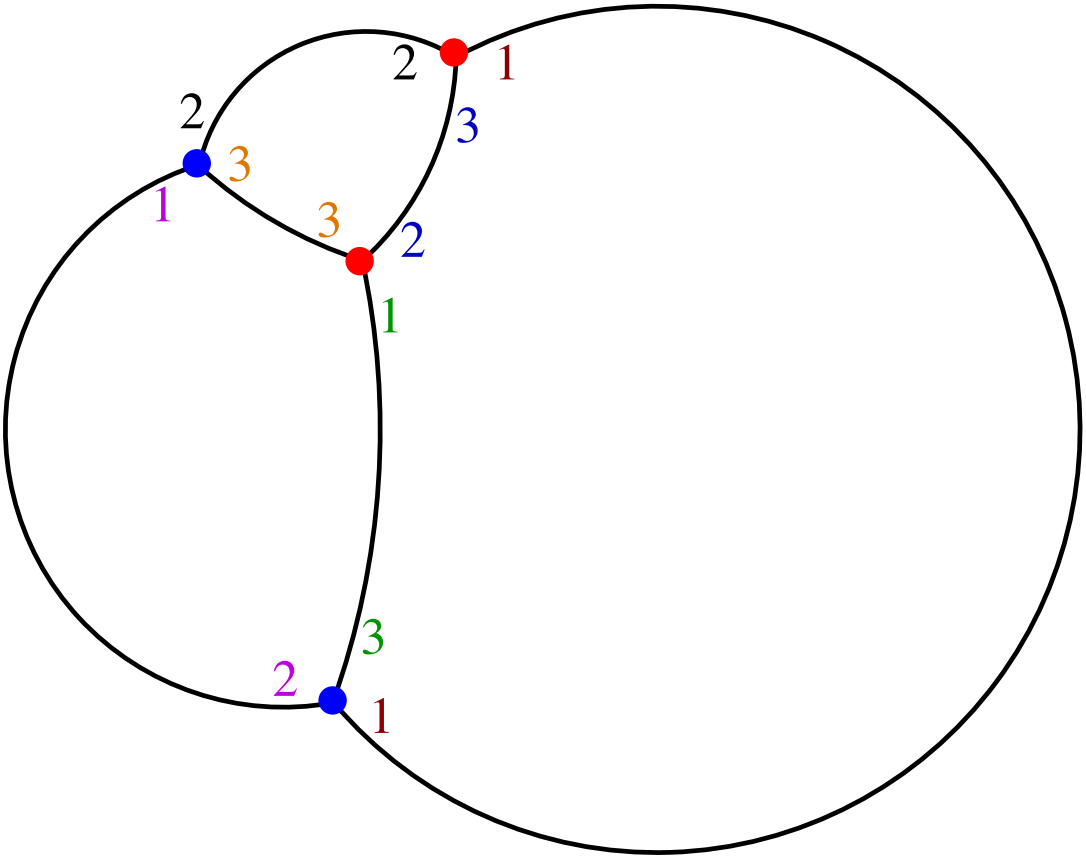

Figure 2: A triple bubble with orientations (same color implies same edge) and parities (see later section, red = +1, blue = -1) marked.

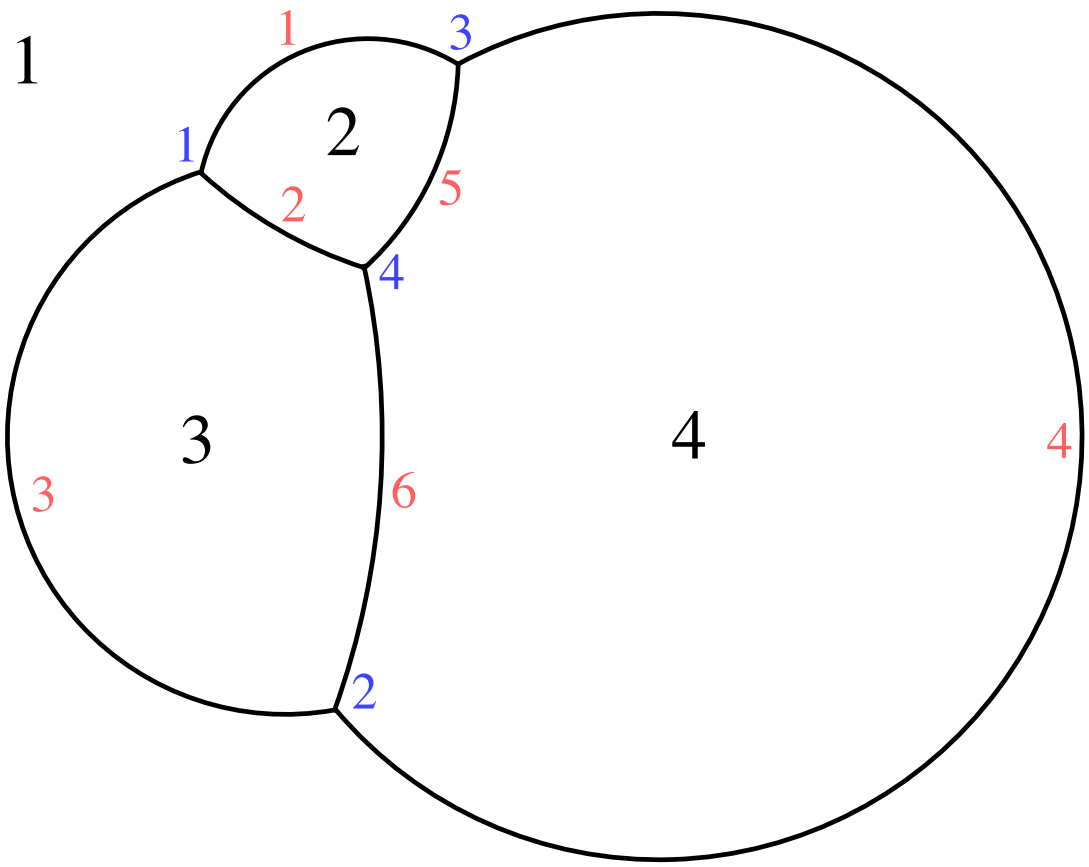

Figure 3: Triple bubble with labels on the faces (black), edges (red), and vertices (blue). The pressures in the faces for the example triple bubble used in this paper are $p_1 = 0, p_2 = 0.426, p_3 = 0.262, p_4 = 0.172$

2 Solving Mancini's equations

In this section I will intersperse the text with references to functions in my Mathematica notebook `20081222mancini.nb`.

The input is a $E \times 2$ matrix `edge` of edges and a $V \times 3$ matrix `vertex` of vertices, as well as a F-dimensional vector `p` of positive real numbers, whose components p_j are the pressure within the jth (interior) face.

The jth row of `edge` consists of the pair of faces (in no particular order) that the jth edge separates. Similarly, the jth row of `vertex` consists of the triple of faces around the jth vertex. We will consider as an example the triple bubble, see fig. 3. In this case, with the faces, edges, and vertices labeled as in the figure (note that the exterior face is given a label too), the matrices are (here written as lists of lists):

$$\texttt{edge} = \{\{1,2\},\{2,3\},\{3,1\},\{1,4\},\{2,4\},\{3,4\}\}$$
$$\texttt{vertex} = \{\{1,3,2\},\{3,1,4\},\{4,1,2\},\{2,3,4\}\}$$

First, using the `p` vector and `edge` matrix we find the absolute pressure differences across each of the edges, `edgep`. The reciprocals of these will yield the radii of curvature of the edges

later (as we'll set the surface tension $\gamma = 1$). This is performed by the function `findedgep` in my Mathematica notebook.

Next, we generate a matrix which gives us the coefficients in Mancini's equations. For later purposes we also need to find the "orientation pair" for each edge, which we described earlier. The function `mancinieqs` takes as input the pressures in the interior faces `p`, as well as `edge` and `vertex` matrices and outputs a matrix `eq` (actually a Mathematica SparseArray construct, which essentially consists of a list of rules defining the nonzero elements of the matrix) such that $\texttt{eq} \cdot (\mathbf{O}, \mathbf{n})_x = 0$ and $\texttt{eq} \cdot (\mathbf{O}, \mathbf{n})_y = 0$ are exactly Mancini's equations. (Here the symbol $(\mathbf{O}, \mathbf{n})_j$ represents a $2V$ column vector whose first V components are the $j = x, y$ component of the quantities $\mathbf{O}$ for each vertex, and whose last V components are the j component of the quantities $\mathbf{n}$ for each vertex). For the triple bubble system that we've been considering, `eq` is:

$$\begin{pmatrix} 1 & 0 & -1 & 0 & C'_{1,1} & 0 & -C'_{1,3} & 0 \\ 1 & 0 & 0 & -1 & C'_{2,1} & 0 & 0 & -C'_{2,4} \\ 1 & -1 & 0 & 0 & C'_{3,1} & -C'_{3,2} & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & C'_{4,1} & -C'_{4,3} & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & C'_{5,3} & -C'_{5,4} \\ 0 & 1 & 0 & -1 & 0 & C'_{6,2} & 0 & -C'_{6,4} \end{pmatrix}$$

The function `mancinieqs` also outputs a $E \times 2$ matrix `orient` whose j,kth component is the orientation I defined earlier on the edge vertex pair corresponding to the jth edge and the kth vertex in the jth row of `edge`. In fact, `mancinieqs` calls a subfunction `Cprime` which, given an index e for an edge and v for one of the vertices, first finds out what orientation the edge has by comparing the pressures, then calculates $C'_{e,v}$ using the formulas above corresponding to each orientation, and returns both $C'_{e,v}$ and the orientation associated to it. Once `mancinieqs` has this information, it simply collects all the C' in matrix form, and the orientations in list-of-pairs form.

Now, all we need to do is solve these equations with the unit vector constraints. Luckily, it turns out that row-reduction of the matrix `eq` greatly simplifies the system. First, note that since the coefficients for the $\mathbf{O}$ variables will always be either ± 1, it should be possible to show that the $\mathbf{n}$ will decouple from $\mathbf{O}$. For instance, in the triple bubble case, the row-reduced matrix for Mancini's equations looks like:

$$\begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 & a_{1,3} & a_{1,4} \\ 0 & 1 & 0 & -1 & 0 & 0 & a_{2,3} & a_{2,4} \\ 0 & 0 & 1 & -1 & 0 & 0 & a_{3,3} & a_{3,4} \\ 0 & 0 & 0 & 0 & 1 & 0 & b_{1,3} & b_{1,4} \\ 0 & 0 & 0 & 0 & 0 & 1 & b_{2,3} & b_{2,4} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Here the $a_{i,j}, b_{i,j}$ are just complicated functions of the $C'_{e,v}$; the second index on these coefficients labels the vertex involved, and $a_{i,j}$ is in the ith linear equation for $\mathbf{O}$ whereas $b_{i,j}$ is in the ith equation for $\mathbf{n}$. Note that the final row vanishes; I believe this is due to the redundancy in $C'_{e,v}$ due to Plateau's rules, but I have not worked this out in general. Indeed, for all the example bubble cluster topologies that I have tried (see figs. 7 - 10),

the row-reduced matrices take this form, where the upper $(V-1)\times V$ block consists of the identity matrix next to a row of -1s, the next $V/2$ rows relate only the $\mathbf{n}$ variables to each other, and the final row is zero. I conjecture that the row-reduced Mancini equations for all topologies can be written in such a "decoupled" form. Intuitively, this must be so, as the coefficients for the $\mathbf{O}$ are all either ± 1.

With this form, if we can solve the equations in $\mathbf{n}$ (now including the unit vector constraints), then by setting say $\mathbf{O}_V = (0,0)$ to use up our translational degrees of freedom, we can solve for all the other $\mathbf{O}$ and be done. Hence we first need to solve the following equations:

$$\mathbf{n}_1 + b_{1,3}\mathbf{n}_3 + b_{1,4}\mathbf{n}_4 = 0$$
$$\mathbf{n}_2 + b_{2,3}\mathbf{n}_3 + b_{2,4}\mathbf{n}_4 = 0$$

If we interpret the coefficients in the equations for $\mathbf{n}$ as lengths, then each of the equations gives a closed polygon in 2D. If we are able to solve for the $\mathbf{n}$ (the directions of the sides) in one equation, we can substitute the known directions into other equations and hence reduce the number of sides of the polygons. I have only been able to solve the equations in the case where at each step, the number of unknown sides in an unsolved equation is at most 2. For instance, in the case of the triple bubble, we start by solving the 5th equation of the row-reduced system. First we use up our rotational degree of freedom by choosing say $\mathbf{n}_2 = (1,0)$. Then, we have quadratic equations in the components of $\mathbf{n}_3, \mathbf{n}_4$. There's a nice geometric interpretation of the two solutions – this is the SSS problem of elementary geometry (with the direction of the $\mathbf{n}_2$ side fixed), and there are two solutions related by a reflection over $\mathbf{n}_2$.

Formulas for this case can be found with computer algebra and are in the expression `solstrip1` in my Mathematica notebook. The functions `solstrip3` and its subroutine `solstrip2` use this expression to solve the triangle case. That is, `solstrip3` takes as input a 2×1 vector A of lengths for the two unknown directions in our triangle and a known vector C which gives the known side of the triangle. Note that aside from using this function to compute solutions to the triangle, I have to solve the nonlinear equations "by hand" by identifying the triangles in the form of the row-reduced matrix and then solving those equations first and then substituting. This should become clear after the next example. This process might be automated as well with some kind of parsing code, but I haven't done it.

Going back to the triple bubble case; we now have two choices of solution for $\mathbf{n}_2, \mathbf{n}_3$. We can solve for $\mathbf{n}_1$ with the fourth equation in the row-reduced system, and then solve for $\mathbf{O}$ as described above, after assuming that $\mathbf{O}_4 = (0,0)$. There's a subtlety here, as after choosing one of the two solutions for $\mathbf{n}_2, \mathbf{n}_3$, $\mathbf{n}_1$ is completely determined by the first equation for the $\mathbf{n}$ – that is, we don't need to use the unit vector condition. However, $\mathbf{n}_1$ always ends up being a unit vector anyways – this degeneracy should be related somehow to Plateau's rules, as in our discussion above, but I haven't worked it out.

Now we have two solution sets for all the $\mathbf{O}, \mathbf{n}$. In general, we expect there to be two solutions at this stage because for every geometric realization of a foam with certain pressures in its cells, the mirror image should also be a realization as well. However, for more complicated topologies, it appears that the number of solutions 2^T (where T is the number of triangle equations solved in the course of finding $\mathbf{n}$) includes mostly spurious solutions. We will discuss this open problem briefly in the last section.

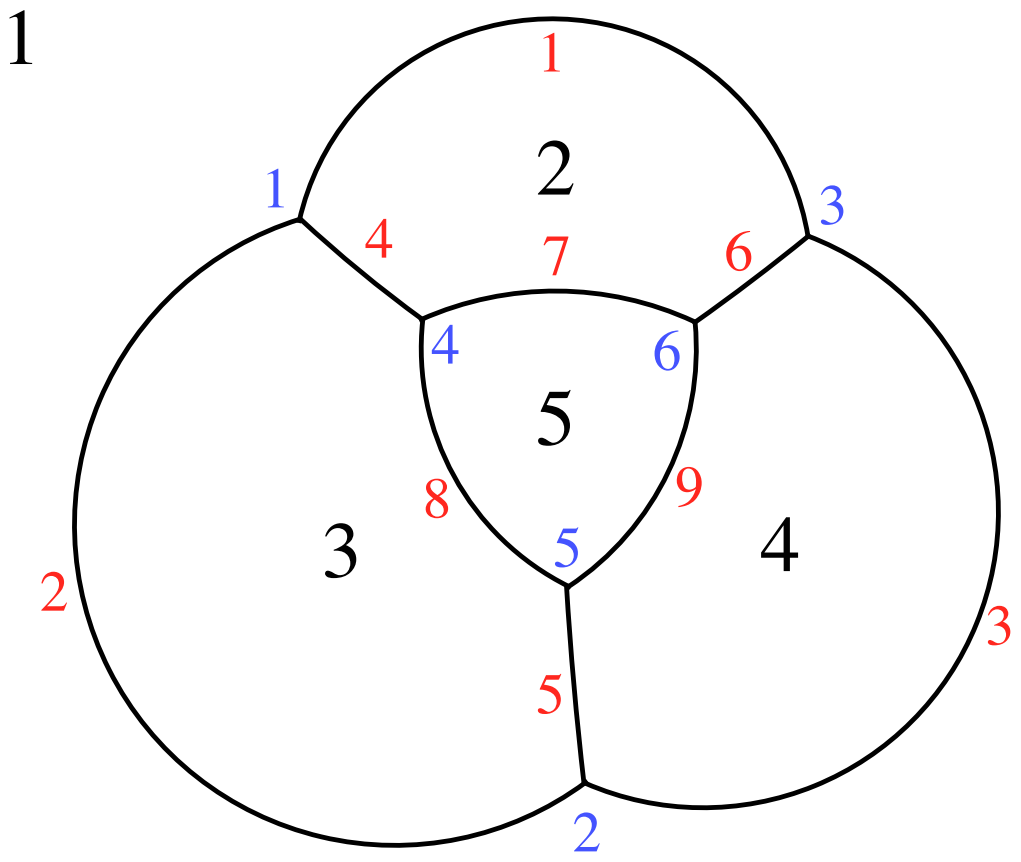

Figure 4: The tripo 4-bubble with labels on the faces (black), edges (red), and vertices (blue).

Here's a more complicated example, the triple bubble with an "o", which is the triple bubble except with another bubble placed at the original center of the triple bubble (alternatively, the triple bubble with a star-triangle move[1] on the center vertex). I call this bubble the "tripo" 4-bubble. See fig. 4, which is a labeled copy of the figure.

Here are the row-reduced Mancini equations for the tripo bubble:

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & a_{1,3} & 0 & a_{1,5} & a_{1,6} \\
0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & a_{2,3} & 0 & a_{2,5} & a_{2,6} \\
0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & a_{3,3} & 0 & 0 & a_{3,6} \\
0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & a_{4,5} & a_{4,6} \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & a_{5,5} & a_{5,6} \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & b_{1,3} & 0 & b_{1,5} & b_{1,6} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & b_{2,3} & 0 & b_{2,5} & b_{2,6} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & b_{3,5} & b_{3,6} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

The heart of this problem is thus to solve the following equations for $\mathbf{n}$:

$$\begin{aligned}\mathbf{n}_1 + b_{1,3}\mathbf{n}_3 + b_{1,5}\mathbf{n}_5 + b_{1,6}\mathbf{n}_6 &= 0\\ \mathbf{n}_2 + b_{2,3}\mathbf{n}_3 + b_{2,5}\mathbf{n}_5 + b_{2,6}\mathbf{n}_6 &= 0\\ \mathbf{n}_4 + b_{3,5}\mathbf{n}_5 + b_{3,6}\mathbf{n}_6 &= 0\end{aligned}$$

To solve this set of equations, we first solve the last equation, which is a triangle, using the formulas in `solstrip3` described above – this yields two solutions for the pair $\mathbf{n}_5, \mathbf{n}_6$. For each of these solutions, we may substitute $\mathbf{n}_5, \mathbf{n}_6$ into the 1st and 2nd equation above:

$$\begin{aligned}\mathbf{n}_1 + b_{1,3}\mathbf{n}_3 + \mathbf{t}_1 &= 0\\ \mathbf{n}_2 + b_{2,3}\mathbf{n}_3 + \mathbf{t}_2 &= 0\end{aligned}$$

Here $\mathbf{t}_j$ are constant vectors depending on the choice we made for $\mathbf{n}_5, \mathbf{n}_6$. We could solve either the 1st or 2nd equation as a triangle; I chose to solve the 2nd. For each $\mathbf{t}_2$, this yields two solutions for $\mathbf{n}_2, \mathbf{n}_3$ – hence we're up to $2 \times 2 = 4$ possible solutions for the $\mathbf{n}$. After this step, the 1st equation is just $\mathbf{n}_1 + \mathbf{s}_1 = 0$ where $\mathbf{s}_1$ is a constant vector arising from substituting $\mathbf{n}_j, j > 1$; note that just as in the triple bubble case, there is a degeneracy in the $\mathbf{n}$ which allows us to find $\mathbf{n}_1$ without using the fact that it's a unit vector. In this bubble $T = 2$, as we were able to solve the equations for $\mathbf{n}$ by applying the triangle formulas twice.

I have been unable to solve the equations corresponding to the foam corresponding to a cube on the sphere stereographically projected to the plane (or indeed any "n-flower" configuration with $n > 3$) as there are no starting triangle equations. The row-reduced matrices in a few of these cases may be found at the end of my notebook – there aren't any triangle equations, and so any solution for the $\mathbf{n}$ may have to come "all at once" to several coupled quadratic equations. It may be the case that these solutions can be simplified via Groebner bases and the like, but I have not tried these techniques yet.

I would like to understand better how the topology of the cluster translates into the structure of the equations for $\mathbf{n}$ after row-reduction – I believe it has something to do with the number of sides around each face, but I have not been able to see anything precise. If we ignore the signs, then the left $E \times V$ submatrix consisting of the coefficients on the $\mathbf{O}$ is an adjacency matrix for the graph of the foam. Presumably something is known about the structure of graph adjacency matrices that can help.

3 Parity and Orientation

In this section I will explain how to use the "orientation" from earlier to partially solve the problem of finding the arcs which define the edges. Fig. 2 shows the orientations on a generic triple bubble configuration. Recall that an edge vertex pair has orientation j when the edge separates the two pressures out of $p_1 < p_2 < p_3$ other than p_j. Consider a vertex v and the three edges e_1, e_2, e_3 where the pair (e_j, v) has orientation j (we argued earlier that the edges meeting in a vertex will all have differing orientations). Note that the edge e_2 must have the greatest pressure difference $k_{31} = p_3 - p_1$ and hence curvature out of e_j. Furthermore, around v, e_1, e_3 curve in the same direction and e_2 curves in the opposite direction. Another way of seeing this is to use the equilibrium condition on the signed curvatures at a vertex: if

$k_{12}+k_{23}+k_{31}=0$ then $k_{31} > k_{12}, k_{23}$ and so must have the opposite sign from them. On the other hand, we cannot say anything about the relative sizes of the curvatures of the e_1, e_3, as either $p_2 - p_1 > p_3 - p_2$ or the opposite could hold.

In order to define the arcs on the circles we derived in the previous section, we must give a starting angle θ_s and an ending angle θ_e when the arc is drawn in a counterclockwise direction. We will now define a parity on the vertices which will allow us to both calculate the endpoint angles of the arcs as well as allow us to decide which is start or end.

We will call the vertex v right-handed, and label it with +1 if when we curl the fingers on our right hand from v outwards along e_2, our thumb points out of the paper, and we'll label v with -1 if our thumb points into the paper. Note the following coincidence – if we draw $\mathbf{n}_v$ from $\mathbf{O}_v$, then if we point our fingers from v to $\mathbf{O}_v$ and curl our fingers in the direction of $\mathbf{n}_v$, our thumb will point out of the board if v is right-handed, and into the board if v is left-handed. In equation form, if $\mathbf{v}$ is the position vector of v, $(\mathbf{O}-\mathbf{v})\times\mathbf{n} > 0$ when v is right-handed and < 0 when v is left-handed. We'll call the ± 1 system the parity of a vertex. However, we cannot find the parity from looking at whether the orientations of the edges around a vertex go $\{1,2,3\}$ counterclockwise or clockwise, for reasons similar to those given at the end of the previous paragraph.

We can use the following set of consistency rules to extend a given parity from one vertex to any other that it is connected to; and thus show that there are two consistent sets of parities on the vertices. See fig. 5 for pictorial derivation of several cases of the consistency rules. The rules may be summarized by saying that if we know the parity of v_1 and v_2 is connected to v_1 by e, then if $\sigma_e = \sum_{1,2} \text{orientation}(e, v_j) \equiv 0 \bmod 2$, i.e. σ_e is even, v_1 and v_2 have opposite parities. If $\sigma_e \equiv 1 \bmod 2$, i.e. σ_e is odd, v_1 and v_2 have the same parity. This simple form of the rule shows that the choice of labels of the orientation was fortuitous.

These rules allow us to extend a parity at one vertex to parities on all the vertices of the foam, however, the correct way to start the process is unknown to me. In the Mathematica notebook, my function `RLmake` takes as input `edge`, `vertex` and `orient`. It then begins by looking at σ_{e_1} where e_1 is the first row of `edge`. If σ_{e_1} is even, a flag variable will assign either the parities (-1,1) to the vertices of e_1 (in that order) or the other choice of (1,-1).

If σ_{e_1} is odd, based on the flag, either the parities (1,1) will be assigned to the vertices of e_1, or the parities (-1,-1). The function then uses the consistency rules above to extend these (guesses for the) parities to the full system. The output of `RLmake` is a $E \times 2$ matrix of the parities such that the jth row consists of the parities of v_1, v_2, where v_1, v_2 are the vertices that the jth edge in `edge` connect.

Fig. 5 illustrates pictoral proofs of rules which allow us to determine which vertex is "start" and which vertex is "end" if arcs are drawn in a counterclockwise direction. The required start / end rules are forced in by the curvature of the arcs, just as the consistency rules were. Let the orientations of (e, v_1) and (e, v_2) be o_1,o_2 respectively. If $o_1 \equiv o_2 \bmod 2$, then by the consistency rules earlier the vertices v_1 and v_2 must have opposite parities. The figure shows that the vertex with parity -1 is start and the vertex with parity +1 is end when $o_1, o_2 \neq 2$; in the case that $o_1 = o_2 = 2$, the vertex with parity +1 is the start and the vertex with parity -1 is the end.

If $o_1 \not\equiv o_2 \bmod 2$, one of o_1, o_2 must be 2, and furthermore, by the consistency rules, the vertices v_1, v_2 have the same parity. If the parities are both +1, the figure shows that the vertex such that $o_j = 2$ must be the start, and the other vertex is the end. If the parities

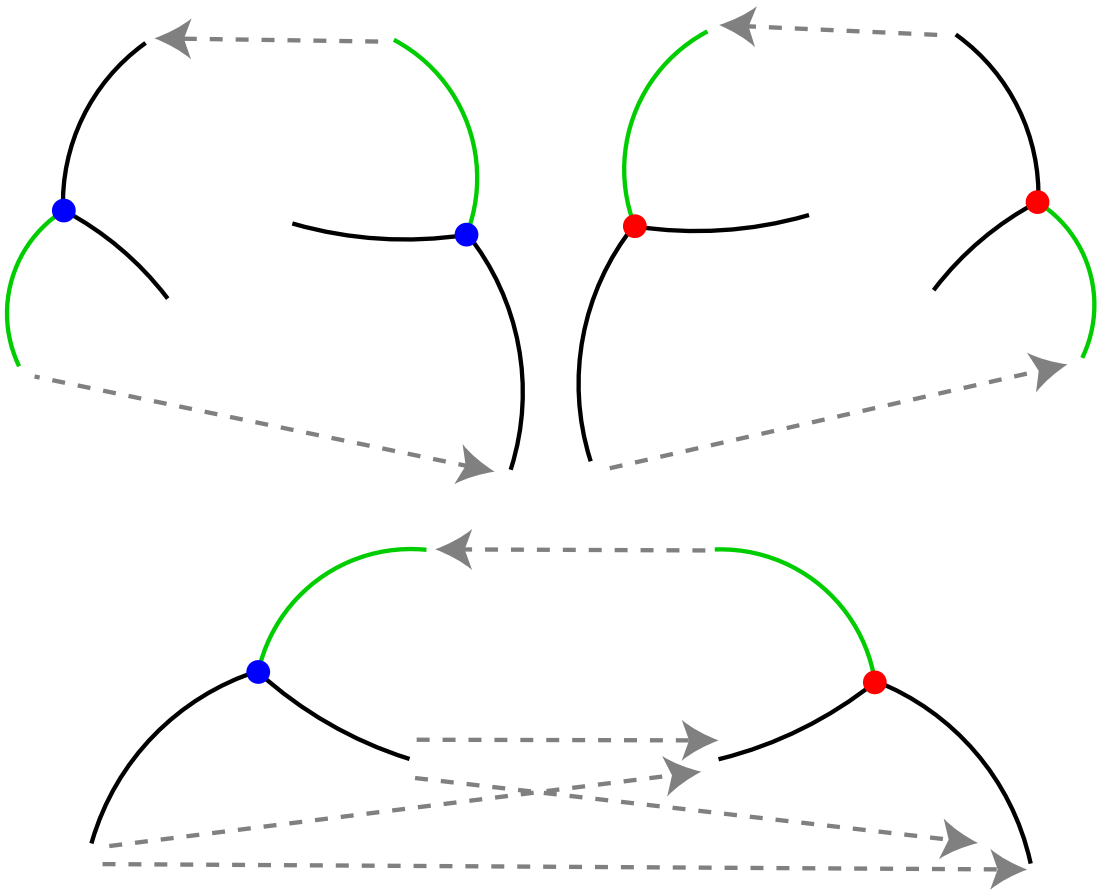

Figure 5: Consistency rules for parities and start / end rules. The proof is simply by enumerating all the ways to consistently connect the arcs with the same curvature direction. The red vertices denote a +1 parity, blue denotes -1. The edge with orientation 2 at each vertex is colored green, and the edges with odd orientation are colored black. Edges which have the right curvature direction and could be connected are connected with a dashed grey line. One can check that when the orientations are both even/odd, the vertices must have opposite parities, and vice versa. The arrows on the dashed grey lines indicate the direction of each arc if it is drawn in a counterclockwise direction. The grey lines thus point from the required start to the end vertices.

are both -1, o_j odd is the start, and $o_k = 2$ is the end.

We may now calculate the two endpoint angles. We can use our earlier rules along with a set of consistent parities on the vertices to decide which is start and end. Given edge e, we have the center and radius of the circle that it lies on. See fig. 6 for the geometry of vertex v on edge e. Since the center of the circle for e is $\mathbf{c}_e = \mathbf{O}_v + C'_{e,v}\mathbf{n}_v$, we see that $C'_{e,v}$ is the distance from $\mathbf{c}_e$ to $\mathbf{O}_v$, and since p_e (the pressure difference across edge e) is the curvature, $p_e C'_{e,v}$ is the cosine of the angle between l_v and the radius of e from $\mathbf{c}_e$ to the position of v. The offset angle α_v is the angle between $-\mathbf{n}_v$ and the x-axis, with the negative sign since $\mathbf{c}_e$ is the origin now, not $\mathbf{O}_v$. Thus the angle θ_v corresponding to v is:

$$\text{atan2}(-(\mathbf{n}_v)_y, -(\mathbf{n}_v)_x) - (\text{parity}_v)\arccos(p_e C'_{e,v})$$

(atan2(y, x) used here yields the angle of the vector (x, y) from the positive x-axis; Mathematica uses a different notation, with the y, x swapped in its `ArcTan` function). Note the factor of minus the parity, due to the placement of $\mathbf{O}_v$ with respect to $\mathbf{c}_e$.

Using this, we can find θ_{v_1} and θ_{v_2} for edge e when it connects vertices v_1 and v_2. We can also use the parity of each vertex to decide which is the start angle and which is the end. In my notebook, the function `thetase` yields a $E \times 2$ matrix of angles defining the arcs of all the edges then. Now the jth row of this matrix consists of θ_s, θ_e (in that order) for the jth edge in `edge`. With this matrix, we can now plot a picture of the foam by drawing in a counterclockwise direction the arcs of the edges from these start angles to the end angles.

4 Results and Outlook

In this section, I'll just describe the results and outlook for this code. There are definitely deeper applications toward the geometry of foams that I have in mind which I have not written up yet and would be glad to discuss. In particular, my goal in writing this program was to explore the "moduli space" or "realization space" of 2D equilibrium foams. The program here allows us, using the sliders, to travel along constant p_j lines in this space. I hope eventually to implement Von Neumann's equations for coarsening on this system (the appropriate coordinates for the moduli space are then the areas of the faces, rather than the pressures), and I welcome any assistance or collaboration with this growing project. I will be speaking about my ideas for such a project at the 2009 APS March meeting.

Figures 7 - 10 are sample images generated by the notebook (in fact, I believe they are images arising from the default values of pressures). For each set of pressures, there are 2×2^T images; the 2^T is due to the quadratic equations we solve when trying to find $\mathbf{n}$. Then, for each of these solutions, there are two choices of parity, since the algorithms above do not know the "right" choice of parity. In the figures, these are divided from each other by a horizontal line. Therefore, in say figure 7, the first image above the line and the first image below arise from the same $\mathbf{O}, \mathbf{n}$, and indeed, the arcs chosen above and below lie on the same circles but are opposite to each other. The same holds true for corresponding images above and below the line in all the figures – they are made of the same circles, but look different because the parities are opposite. The figures are not set to be drawn at the same scale in the current code – rather Mathematica resizes each image so that it best fits in the space in the grid.

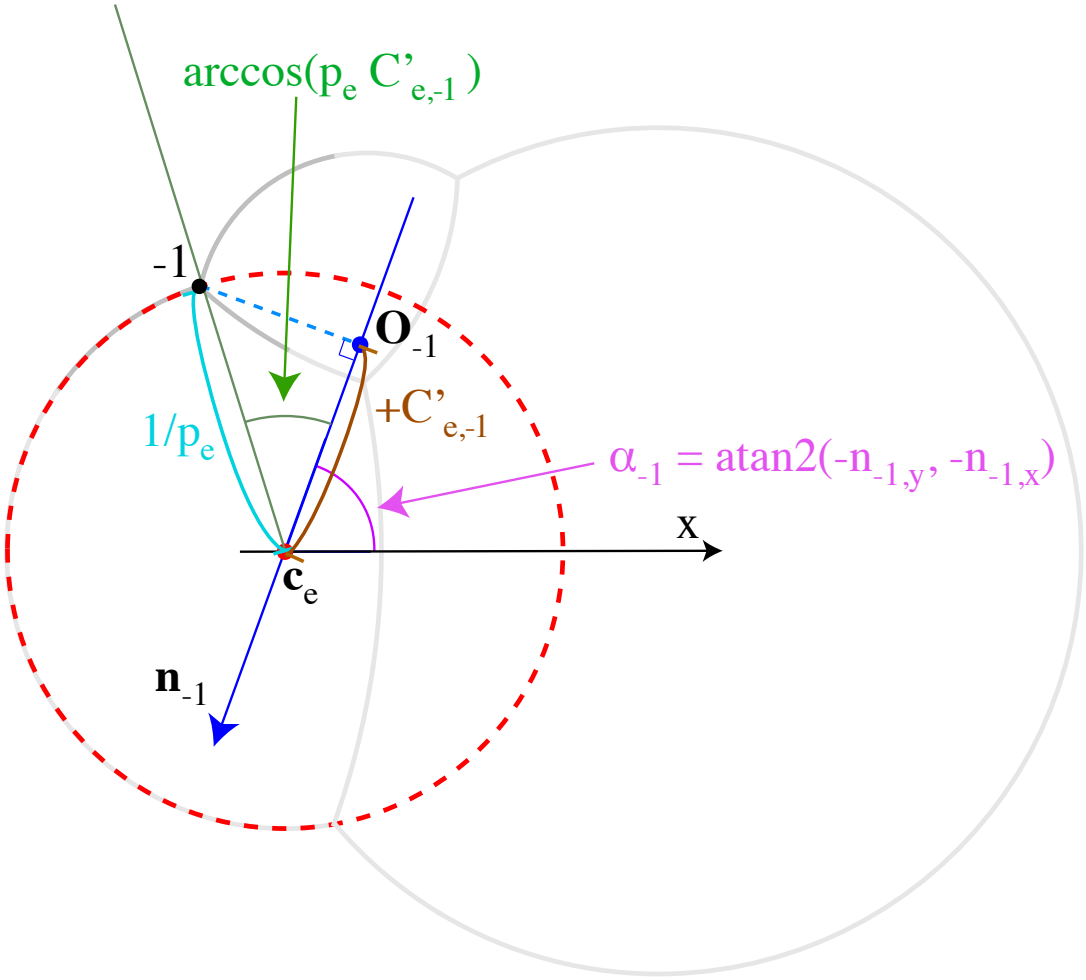

Figure 6: Figure for determining the angles that the vertices are at on a given edge circle. The vertex shown here has parity -1, so we've named it "-1". The figure expresses the fact that $\theta_{-1} = \alpha_{-1} + \arccos(p_e C'_{e,v})$. Note that $C'_{e,-1} > 0$ since $\mathbf{n}_{-1}$ points from $\mathbf{O}_{-1}$ to $\mathbf{c}_e$.

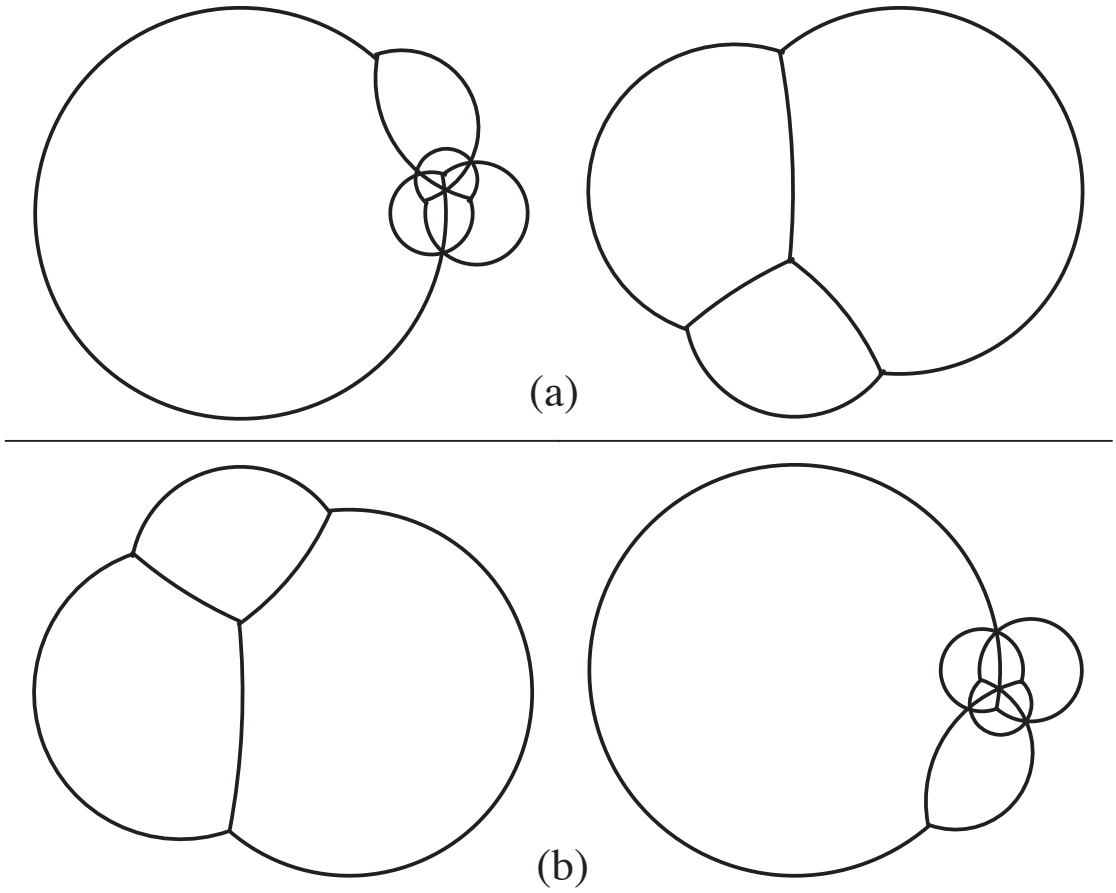

Figure 7: Triple bubble output from notebook with (a) choice 1 for parity, (b) choice 2 for parity. Note that different images are not drawn to the same scale. The corresponding images above and below the horizontal line arise from the same circles (i.e. same choice of solution for $\mathbf{O}, \mathbf{n}$), but differ by the choice of parity.

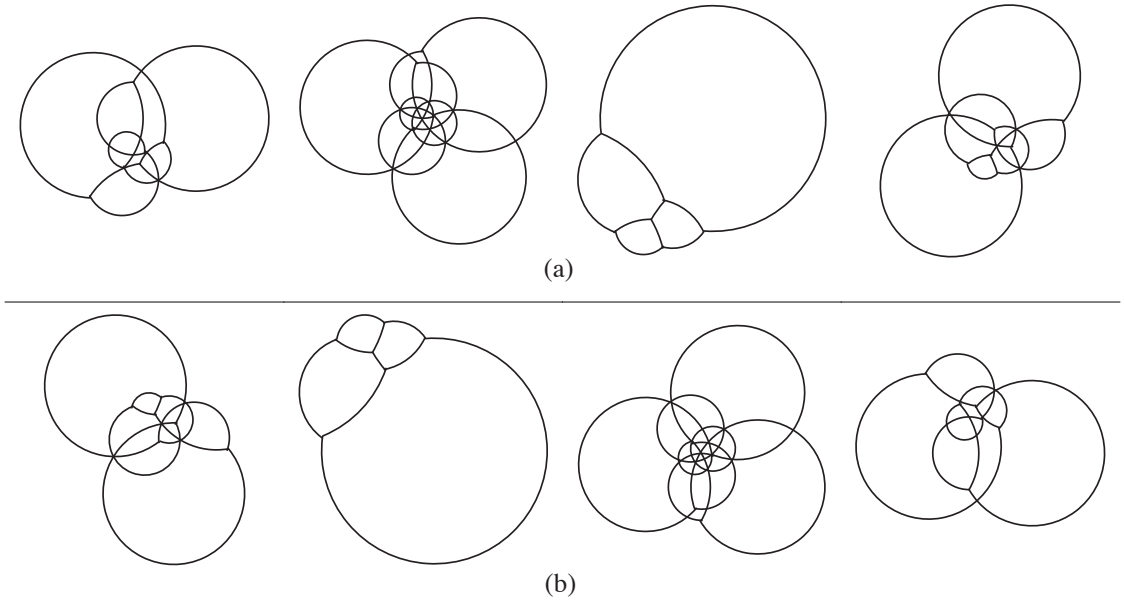

Figure 8: "X" 4-bubble output with (a) choice 1 for parity, (b) choice 2 for parity. Note that different images are not drawn to the same scale. The corresponding images above and below the horizontal line arise from the same circles (i.e. same choice of solution for $\mathbf{O}, \mathbf{n}$), but differ by the choice of parity.

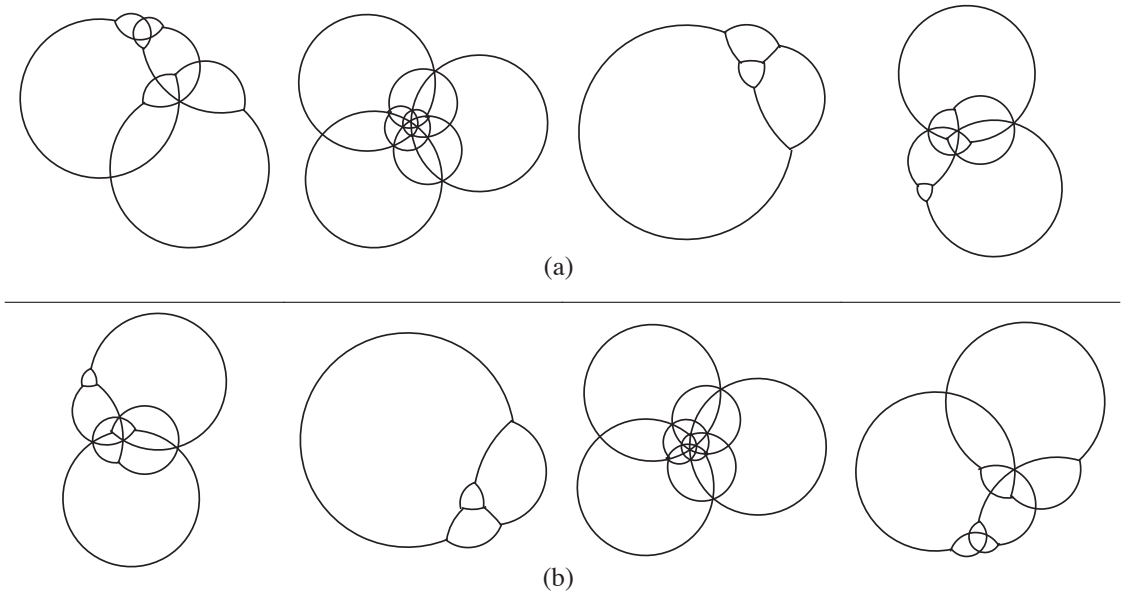

Figure 9: Tripo 4-bubble with (a) choice 1 for parity, (b) choice 2 for parity. Note that different images are not drawn to the same scale. The corresponding images above and below the horizontal line arise from the same circles (i.e. same choice of solution for $\mathbf{O}, \mathbf{n}$), but differ by the choice of parity.

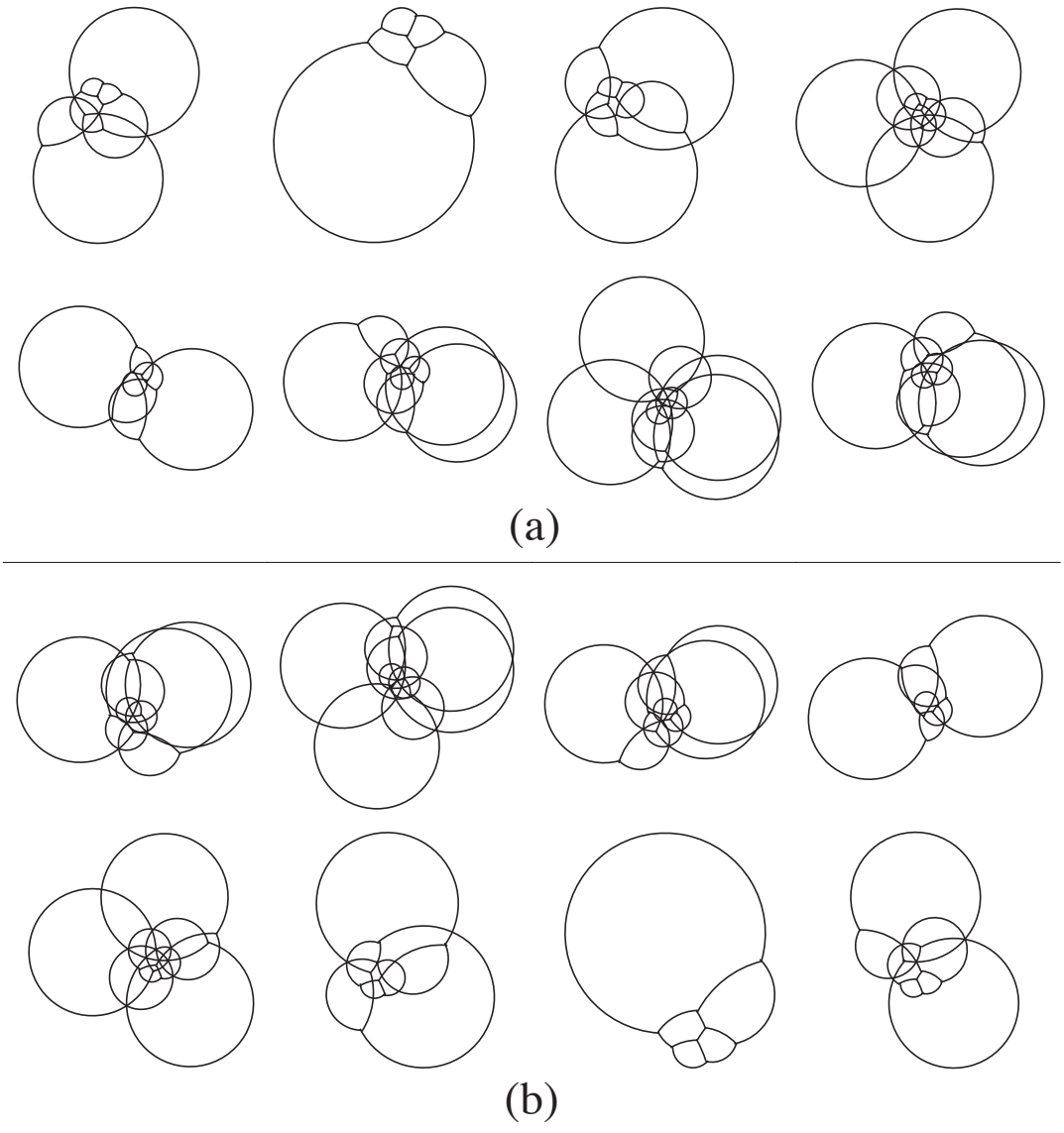

Figure 10: Bubble with the same topology as Figure 2.9 in [1] with (a) choice 1 for parity, (b) choice 2 for parity. Note that different images are not drawn to the same scale. The corresponding images above and below the horizontal line arise from the same circles (i.e. same choice of solution for $\mathbf{O}, \mathbf{n}$), but differ by the choice of parity.

The choices of parity are called choice 1 and choice 2 in the figure captions, and obviously, this labeling of the parities is non-canonical, as it just describes a certain choice for the starting parity at the first vertex in the list `vertex`. This is obviously a rather serious gap in my understanding of the geometry of foams at this point.

With a given choice for parity, only one of the pictures "looks right", that is, there are no intersections of the edges except at vertices. In fact, two of the $\mathbf{O}, \mathbf{n}$ choices are correct, but only one ends up looking right for each choice of parity. If we change the arbitrary choice to the opposite one, then another picture looks right, indeed the two valid pictures are mirror images of each other (compare a/b in any of the figures). What this says is that not all of the solutions to Mancini's equations yield valid foams, at least without more information. The issue of which choice of $\mathbf{O}, \mathbf{n}$ therefore needs to be addressed before more complicated foams can be calculated, as is evident from fig. 10 (a bubble topology taken from figure 2.9 in [1]), where there are 8 solutions to Mancini's equations, and only 2 solutions give valid foam structures (with the right choice of parity, that is).

If we vary the pressures in the faces through points where there are equal pressures in adjacent faces, the good choices of $\mathbf{O}, \mathbf{n}$ also jumps between different choices of quadratic roots. (Note that since I have not implemented the correct equations for the case when adjacent pressures are equal (which are in [1]), I have included code that adds some slight numerical randomness to the chosen pressures so that the code does not fail when sliding through such points). In fact, at points where adjacent pressures are equal, I believe that the quadratic equations have double roots, and therefore the system is degenerate. An extension of the ideas of orientation and parity to the case with degenerate pressures ought to allow us to connect the orientations and parities in one choice of roots to another as we vary the pressures through those degenerate points. I don't quite have a full understanding of this set of affairs yet either.

There are also regions of p_j space where no foams exist at all. For instance, take $p_1 \approx p_2 \approx p_3 \approx 0.2, p_4 = 0.3$ in the tripo 4-bubble configuration, then decrease the pressure of p_4. No valid foams exist in these regions because some of the edges have vanished – this is representative of so-called T_1 events in foams. It would be nice to see if there's an analytic description of these "boundaries" of the 2D foam moduli space.

There are thus three problems yet to be solved before the Mancini program is completed. First, we need a way to solve arbitrarily complicated coupled equations for $\mathbf{n}$. This may be out of reach, at least analytically. Understanding how the structure of the row-reduced Mancini's equations arises from the topology of the graph of the foam will definitely help. Second, we need a way to identify the correct parity choice for a picture, instead of just making an arbitrary choice. Third, we need a way to identify the choices of solution which do not self-intersect. It is possible that the only way to accomplish the second and third problems is to check each pair of arcs to see whether they intersect at points other than vertices, but this would still require checking all 2^T choices – it's my hope that there's a more fundamental way of seeing this. I welcome any thoughts towards solving these problems.

Acknowledgments. I thank Randall D. Kamien, Carl Modes, and Gareth Alexander for productive and helpful conversations. I also thank Marco Mancini for an inspiring thesis. Support from NSF, Bernstein, Coburn.

References

[1] M. Mancini. Structure and Evolution of Soap-Like Foams, 2005. Available at http://tel.archives-ouvertes.fr/tel-00010304/fr/