# Reconstruction of Two-Dimensional Foams

Brian Foley - brianf@maths.tcd.ie - 97066231

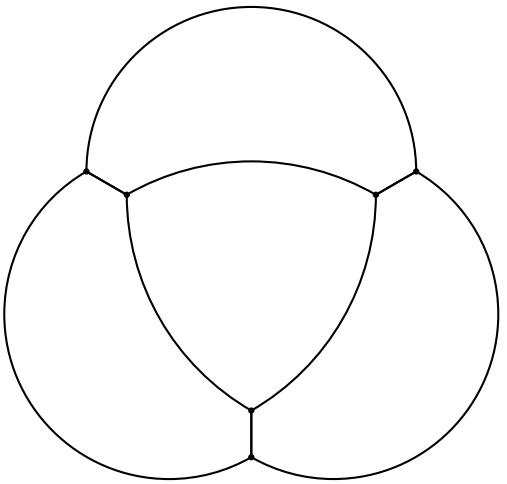
Advisor: Stefan Hutzler

Michelmas Term, 2001

**Abstract**

The experimental determination of certain properties of two-dimensional foams at equilibrium is a difficult problem. In particular, determining the internal pressures of bubbles and determining the exact position of bubble walls is of interest to us. There is no way to know the internal pressures without intrusive measurements and accurately digitising the location of the walls (especially if the foam is very wet) is almost impossible.

This project aims to fully recreate a two-dimensional foam cluster, including the bubble pressures and wall curvatures, given the topology and the location of the vertices of a small foam cluster in equilibrium.



# Contents

# List of Figures

# 1 Introduction

Examine the experimentally generated foam in Figure (1). It is quite difficult to determine the curvature of the walls in the foam because the walls are very thick, which, coupled with a large radius of curvature makes any curve fitting quite prone to errors. It is equally difficult to calculate the internal pressures of the bubbles without invasive measurements that would disturb the bubble structure. However it is relatively easy to determine the coordinates of the vertices, either manually or computationally, because, even though the top sheet of glass containing the 2-d foam may be quite wet, it is still possible to distinguish the narrowest part of the vertical junction of the three walls. This point is a slightly darker colour than the rest of the walls in Figure (1), for example.

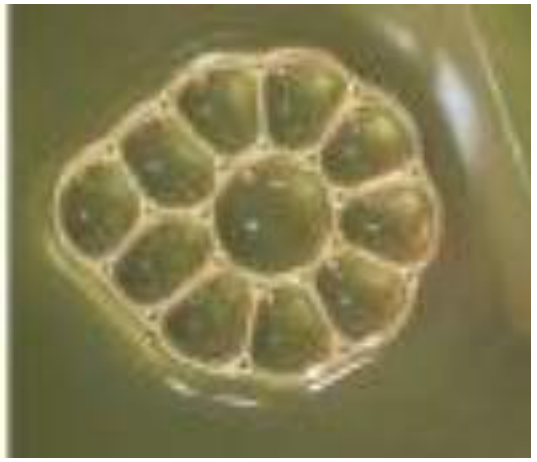


Figure 1: An experimentally generated foam

Thus, the motivation behind this project is to find an easy way to get numerical curvature and pressure data from experimentally or computationally generated images, where it is assumed that the user can somehow determine both the topology and vertex locations from the foam using other means.

# 2 Theory

## 2.1 General

Consider a bubble cluster in equilibrium. One face of one bubble in that cluster is shown in Figure (2).

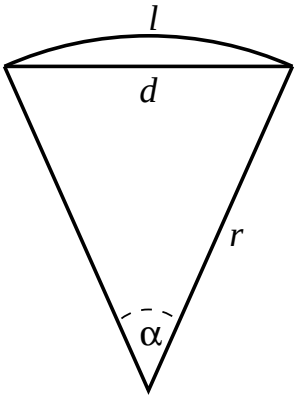



Figure 2: One wall of a bubble cluster, $l$ is the curved length, $d$ is the straight-line distance between two vertices, $r$ is the radius of curvature and $\alpha$ is the angel of the arc.

Clearly the angle $\alpha$ can be given by:

$$\alpha = \frac{l}{r}$$

Thus the turning angle of the tangent to the wall bubble along the entire length $l$ is also $\alpha$. We know that the internal angles at each vertex are $120^{\text{O}}$, thus we can say that the sum of the turning angles for any particular bubble is given by:

$$\sum_{i=1}^{n} \frac{l_i}{r_i} + n\frac{\pi}{3} = 2\pi$$

Express the radii of curvature as a function of the pressure difference:

$$r_{ij} = \frac{2\mu}{p_i - p_j}$$

Which gives:

$$\sum_{i=1}^{n}(p_i - p_j)l_{ij} = \frac{2\pi}{3}\mu(6 - n_i) \tag{1}$$

Some simple geometry yields the equation:

$$l_{ij} = r_{ij} 2sin^{-1}\left(\frac{d_{ij}}{2r_{ij}}\right)$$

## 2.2 Implementation

The general procedure to find the exact form of any foam cluster is as follows:

- Initialise $l_{ij}$'s to be the straight line lengths between the two vertices.
- Solve eqn (1) for $P_i$.
- Find $r_{ij}$'s from $P_i$'s.
- Get $l_{ij}$'s from $r_{ij}$'s.
- Start again, but this time with the newly-calculated values for $l_{ij}$.
- Continue until the energy has come to equilibrium.

Here we present an example of this procedure for a simple, symmetric, four bubble cluster:

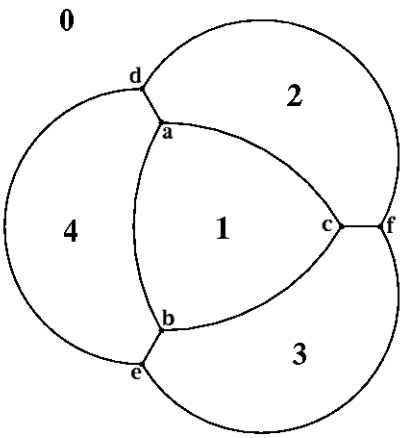



Figure 3: A symmetric four-bubble cluster

Use the equation:

$$\sum_{i=1}^{n}(p_i - p_j)l_{ij} = \frac{2\pi}{3}\mu(6 - n_i)$$

for each bubble above to get the equations:

$$
\begin{aligned}
(p_1 - p_2)l_{12} + (p_1 - p_3)l_{13} + (p_1 - p_4)l_{14} &= \frac{2\pi}{3}\mu(6-3) \\
(p_2 - p_0)l_{20} + (p_2 - p_1)l_{21} + (p_2 - p_3)l_{23} + (p_2 - p_4)l_{24} &= \frac{2\pi}{3}\mu(6-4) \\
(p_3 - p_0)l_{30} + (p_3 - p_1)l_{31} + (p_3 - p_2)l_{32} + (p_3 - p_4)l_{34} &= \frac{2\pi}{3}\mu(6-4) \\
(p_4 - p_0)l_{40} + (p_4 - p_1)l_{41} + (p_4 - p_2)l_{42} + (p_4 - p_3)l_{43} &= \frac{2\pi}{3}\mu(6-4)
\end{aligned}
$$

This reduces to:

$$
\begin{pmatrix}
0 & L_1 & -l_{12} & -l_{13} & -l_{14} \\
-l_{20} & -l_{21} & L_2 & -l_{23} & -l_{24} \\
-l_{30} & -l_{31} & -l_{32} & L_3 & -l_{34} \\
-l_{40} & -l_{41} & -l_{42} & -l_{43} & L_4
\end{pmatrix}
\begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix}
= \mu
\begin{pmatrix} 2\pi \\ \frac{4\pi}{3} \\ \frac{4\pi}{3} \\ \frac{4\pi}{3} \end{pmatrix}
\tag{2}
$$

where $L_i$ is the total boundary length of bubble $i$. Now we set $P_0 = 0$ and $\mu = 1$ to get:

$$
\begin{pmatrix}
L_1 & -l_{12} & -l_{13} & -l_{14} \\
-l_{21} & L_2 & -l_{23} & -l_{24} \\
-l_{31} & -l_{32} & L_3 & -l_{34} \\
-l_{41} & -l_{42} & -l_{43} & L_4
\end{pmatrix}
\begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix}
=
\begin{pmatrix} 2\pi \\ \frac{4\pi}{3} \\ \frac{4\pi}{3} \\ \frac{4\pi}{3} \end{pmatrix}
\tag{3}
$$

this is solvable for $P_i$, and we can use:

$$
r_{ij} = \frac{2\mu}{p_i - p_j}
$$

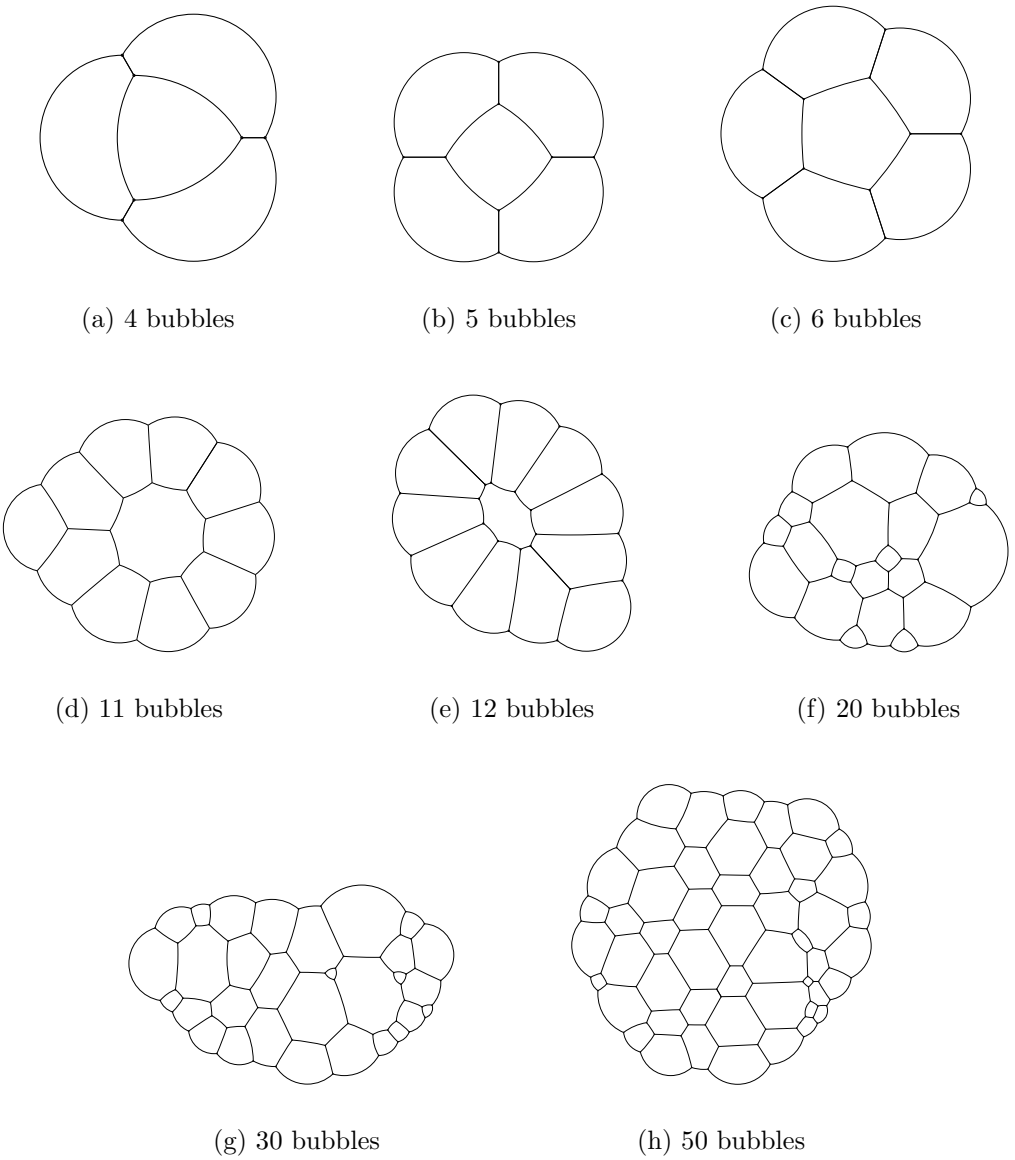
to get $r_{ij}$, which we substitute into:

$$
l_{ij} = r_{ij} 2sin^{-1}\left(\frac{d_{ij}}{2r_{ij}}\right)
$$

to get $l_{ij}$, which is the curved wall lengths. We continue in this fashion until the total wall length has been minimised.

# 3 Results

## 3.1 Overview



(a) 4 bubbles (b) 5 bubbles (c) 6 bubbles

(d) 11 bubbles (e) 12 bubbles (f) 20 bubbles

(g) 30 bubbles (h) 50 bubbles

Figure 4: A summary of some of the results. Figure 4(d) was created using experimentally determined vertices, all the other clusters were generated from vertices provided by Simon Cox from his *Surface Evolver* simulations. Larger versions of all these images are provided in the appendices for closer examination.

Typical outputs from my program are shown in Figure (4). Supplementary output is also printed to the screen in the form of the final calculated pressures of the bubbles. There are also several different outputs that can be used for debugging purposes such

as printing the initial conditions and printing the bubble index numbers in the final output.

## 3.2 Bubble Pressures

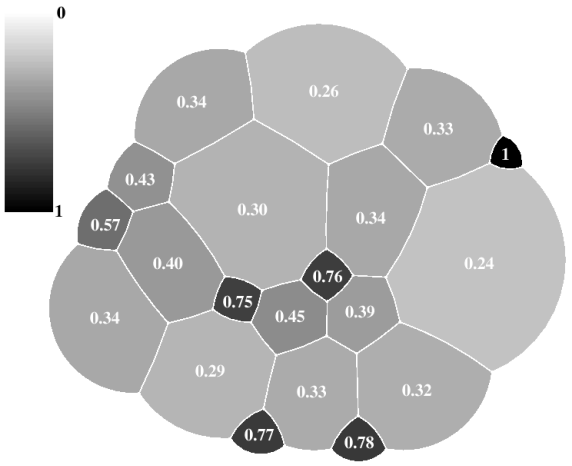



Figure 5: Twenty bubble cluster showing relative normalised pressures

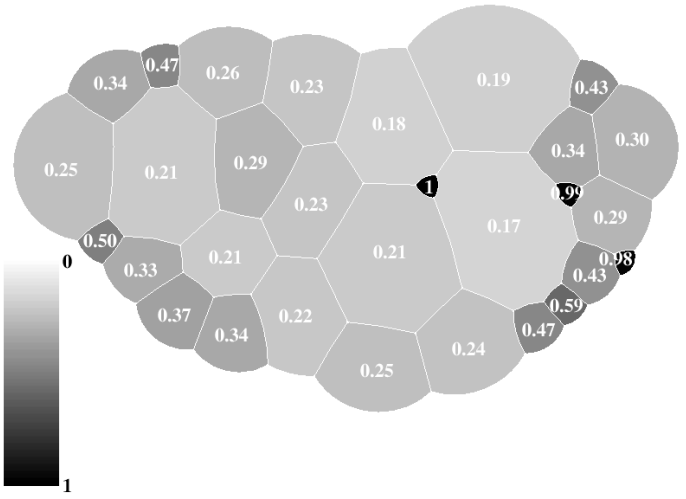



Figure 6: Thirty bubble cluster showing relative normalised pressures

Bubble pressures are calculated during the course of the calculation. Figures (5), (6) & (7) show twenty, thirty & fifty bubble clusters respectively and the normalised pressures of each individual bubble. There are several features visible in these diagrams that one would expect to see. Firstly, in the fifty bubble cluster there is a large area of six-sided bubbles in the center that all have the same pressures. This is to be expected

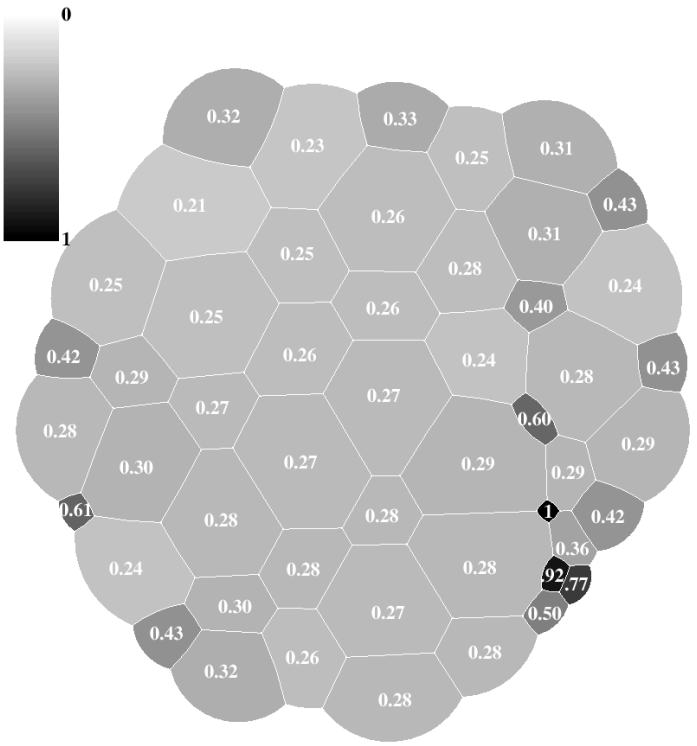



Figure 7: Fifty bubble cluster showing relative normalised pressures

as hexagons are the natural arrangement for bubbles in equilibrium. Towards the edge of this cluster there are many smaller bubbles, all of which have higher pressures. It is clear that the six-sided bubbles have lower pressures than those bubbles that have a lower number of sides.

## 3.3 Comparisons

### 3.3.1 Bubble Shapes

We can compare the generated bubble shapes with two other types of data, namely experimental data and simulated data. Figure (8) shows reconstructed data superimposed on top of experimental data. The vertices used for the reconstruction were acquired by digitising the experimental image. As you can see the images appear to be identical in terms of position and curvature of the walls. Figure (9) compares a different reconstructed foam with the computationally generated structure that its vertices were derived from. Once again, it is clear that the two images are almost identical, with any minor differences so small as to be undetectible.

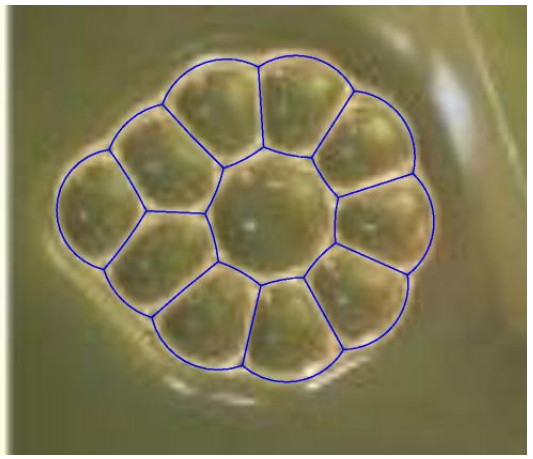


Figure 8: Image of bubble cluster shown in Figure (1) on page 1, with the reconstructed structure superimposed in blue.

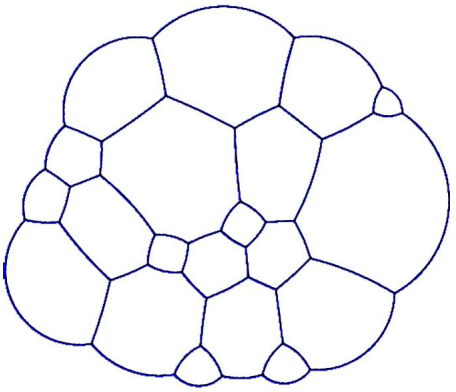


Figure 9: A comparison of Simon Cox's *Surface Evolver* calculation (in blue) and my bubble structure (in black) generated from vertices taken from his calculation. The line widths are almost identical and you can see that the blue lines are almost totally obscured.

### 3.3.2 Bubble Pressures

Figure (10) compares the reconstructed pressures with the pressures calculated by Simon Cox using *Surface Evolver*. The differences in pressures detected are mostly less than 0.001% of Simon Cox's calculated pressures. This number is so low as to be almost negligible. However, it should be noted that bubbles 13, 17 & 19 have the highest percentage difference. If you examine the bubble-numbering scheme for this cluster given in Figure (19) on page C-6 in Appendix C, you will note that these bubbles are the smallest bubbles in the cluster (apart from bubble 15, which was the smallest, but was used for normalising, thus having difference zero). This points to growing loss of accuracy for small bubbles in either my system or *Surface Evolver's* calculations when

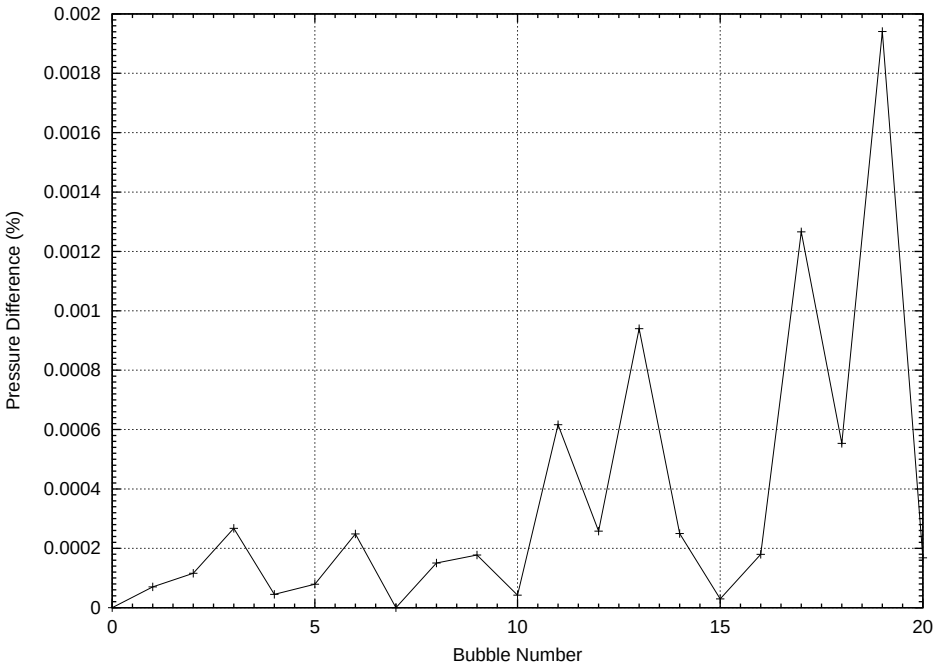



Figure 10: Pressure Differences

dealing with small bubbles.

## 3.4 Non-equilibrium initial conditions

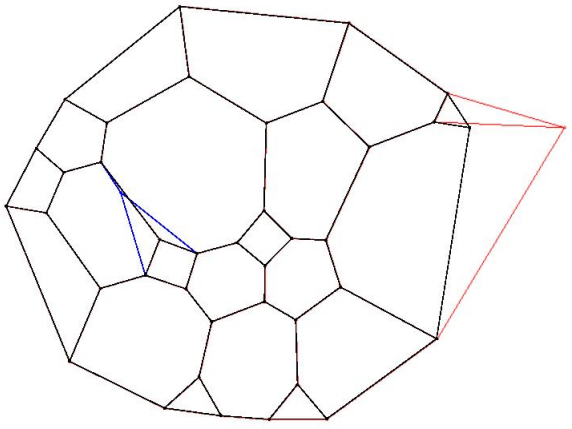


Figure 11: Inital conditions for non-equilibrium conditions. The black lines represent the equilibrium conditions, the red lines show a point on the outside being displaced and the blue lines show a point on the interior being displaced. One point has been moved in each case.

Figure (11) shows a twenty bubble cluster (in black) with two non-equilibrium clusters (in red and blue) superimposed on the top. It was found that the program will

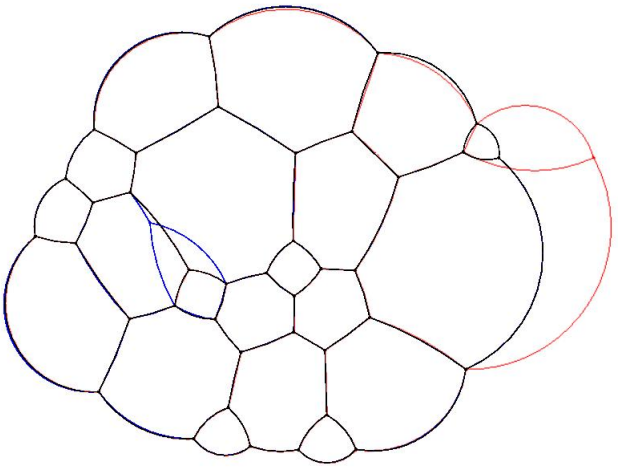


Figure 12: Reconstructed foams for non-equilibrium conditions. The black lines represent the equilibrium conditions, the red lines show a point on the outside being displaced and the blue lines show a point on the interior being displaced. It is apparent that the program will do its best to form a foam given the initial veritces, if those vertices do not represent an equilibrium foam then the program will do the best it can.

operate even when the vertices fed to it are non-equilibrium veritces. Figure (12) shows the results. In both cases, most of the cluster was rendered correctly, with some minor defects close to the point of error. In the blue case, where an internal vertex was moved a small distance, there is only noticable differences around the vertex and in the nearest boundary bubble. In the red case, where an external vertex was moved a larger distance, there is a difference on two of the boundary bubbles and around the vertex.

Given the distances that the vertices were moved, it is clear that this reconstruction is remarkably tolerent to defects in the vertices positions if only a small number of vertices are incorrect. The "damage" caused by an incorrect vertex will always be localised and will not tend to spread throughout the foam to a large extent.

# 4 Limitations

## 4.1 Finite Clusters

This method of reconstruction is limited to finite clusters of bubbles, *i.e.* it is not able to reconstruct a random sector of a larger foam. The logic used in this reconstruction technique depends in part on the fact that outside the area we are interested in the pressure is zero. This would obviously not be true if we looked at a section of a larger foam.

## 4.2 Small Clusters

This program cannot deal with small clusters. It will not draw any *long arc* bubbles, *i.e.* it will not draw any arcs that bulge out more than $180^{\mathrm{o}}$. Also it cannot handle the single bubble case because it does not have any vertices, and thus the turning angles cannot be calculated.

## 4.3 Generation of input files

There is another very real limitation of the usefulness of this reconstruction technique. It is very tedious to generate the input files for this program. Most of the difficulty with this input is that specifying the topology of the foam is a complex (and long) task. When this was done by the author during the author during the completion of this project, the largest foam used was a fifty bubble cluster, this cluster took on the order of 30 minutes to digitise and to generate the input file. It is clear then that for really big clusters the time needed to generate the input file will increase dramatically and will be prone to errors. There is a debugging command-line option to help you check for mistakes in the input files (**-check**). Furthermore the input file format is explained in detail in Appendix B.

# 5 Conclusion

In this paper I have shown that it is possible to reconstruct a foam from two initial conditions; topology and vertex locations. I have sucessfully reconstructed a range of experimentally and computationally generated images, and I have shown that these reconstructions agree with all the experimental and computational data available to us. Also I have shown that errors in the input coordinates will only result in local mistakes in the foam and that these errors will not affect the foam as a whole.

# References


[1] D. L. Weaire and Stefan Hutzler *The Physics of Foams*, Oxford, Clarendon Press, 1999.

[2] Simon Cox *Private communication*, Sundry computational and experimental images of foam used in this project.

# Appendix

# A Code

The following code is the function that does all the work in this program. The while loop on line 18 will continue to adjust the pressures and radii of curvature until the energy has been minimised or a certain number of iterations have been reached, whichever comes first.

```
calculate(int *n, int **array, double ***coords, int ***registry, double
        ***straight_length, double ***curved_length, double
        ***new_curved_length, double ***radius_of_curvature, double **C, double
        **P, double ***M, double *energy, double *old_energy, double *tolerance,
5       double *weight, double *iterations){

    int i, j, k, l, count=0;

    k = (*array)[0];
10
    for(i=0; i<=(*n); i++){
        for(j=i; j<=(*n); j++){
            *energy += (*curved_length)[i][j];
        }
15  }
    printf("0: %lf\n", (*energy));

    while(count<(*iterations) && (fabs(*energy-*old_energy)>(*tolerance))){
        count++;
20      /* create the rhs of the matrix */
        for(i=(k-*n-1);i<k;i++){
            if((*coords)[i][0] != 0){
                (*C)[((int)(*coords)[i][0])-1] = ((2.0*PI*MU)/3.0)*(6-((int)(*coords)[i][1]));
            }
25      }

        /* create the M matrix */
        for(i=0; i<(*n); i++){
            for(j=0; j<(*n); j++){
30              (*M)[i][j] = 0.0;
                if(i == j){
                    for(l=0; l<=(*n); l++){
                        (*M)[i][j] += (*curved_length)[i+1][l];
                    }
35              }
```

```
            else{
                (*M)[i][j] = -(*curved_length)[i+1][j+1];
            }
        }
    }

    /* do the matrix solving */
    gaussian_elimination(&*M, &*C, *n);
    back_substitution(&*M, &*C, &*P, *n);

    /* re-arrange P to account for P[0] the vacuum pressure */
    for(i=(*n); i>=1; i--){
        (*P)[i] = (*P)[i-1];
    }
    (*P)[0] = 0;

    /* calculate the radii of curvature */
    for(i=0; i<=(*n); i++){
        for(j=0; j<=(*n); j++){
            if((*registry)[i][j] == 1){
                (*radius_of_curvature)[i][j] = (2.0*MU)/((*P)[i]-(*P)[j]);
                if((*radius_of_curvature)[i][j] < -BIG){
                    (*radius_of_curvature)[i][j] = -BIG;
                }
                if((*radius_of_curvature)[i][j] > BIG){
                    (*radius_of_curvature)[i][j] = BIG;
                }
            }
        }
    }

    /* calculate curved lengths */
    for(i=0; i<=(*n); i++){
        for(j=0; j<=(*n); j++){
            if((*registry)[i][j] == 1){
                if(fabs((*radius_of_curvature)[i][j]) > (BIG-1)){
                    (*curved_length)[i][j] = (*straight_length)[i][j];
                }
                else{
                    if((*straight_length)[i][j]/(2*fabs((*radius_of_curvature)[i][j])) > 1.0){
                        (*new_curved_length)[i][j] = (PI*((*straight_length)[i][j]))/2.0;
                    }
                    else{
                        (*new_curved_length)[i][j] = fabs((*radius_of_curvature)[i][j])
                            *2*asin(((*straight_length)[i][j])/(2*fabs(((*radius_of_curvature)[i][j]))));
```

```
                    }
                    (*curved_length)[i][j] = ((*weight)*((*curved_length)[i][j]))
                        +((1.0-(*weight))*((*new_curved_length)[i][j]));
                }
            }
        }
    }

    *old_energy = *energy;
    *energy = 0.0;

    for(i=0; i<=(*n); i++){
        for(j=i; j<=(*n); j++){
            (*energy) += (*curved_length)[i][j];
        }
    }

    printf("%i: %lf\n", count+1, (*energy));
  }
}
```

# B Sample Input File

Below is a sample input file, for the four-bubble cluster shown in Figure (3).

```
-0.343962759785536 0.595674619746094
-0.343969727338344 -0.595701347379678
0.687796925645181 -0.00001248452384
-0.45531647990125 0.788546609891099
-0.455321477622829 -0.788565965092777
0.910500891276261 -0.00001217294436

0 3 1 5 4 2 4 6 4 6 5
1 4 0 5 4 2 4 1 3 1 2 4 2 5
2 4 0 4 6 4 6 3 3 3 1 1 1 4
3 3 2 1 3 4 3 2 1 2 1
4 4 3 2 3 2 3 6 0 6 5 1 5 2
```

The file should have a series of lines at the top with two numbers per line, beneath that there should be a blank line followed by several lines, each one specifying the topology of each bubble. Each of these vertex lines has $2 + (3 \times n)$ numbers where $n$ is the number of neighbours that the particular bubble has. Bubble 0 must be the outside bubble. The first number is the number assigned to each bubble, the second number is the number of sides the bubble has, and the remaining groups of three are in the format "neighbour, vertex number, vertex number", which simply tells you the two vertices that are joined together to form the wall that separates the bubble and its neighbour.

# C Full-size versions of images

Below I have reproduced full-sized versions of all the images used in this report. These are provided to enable closer examination of the structures, if necessary.

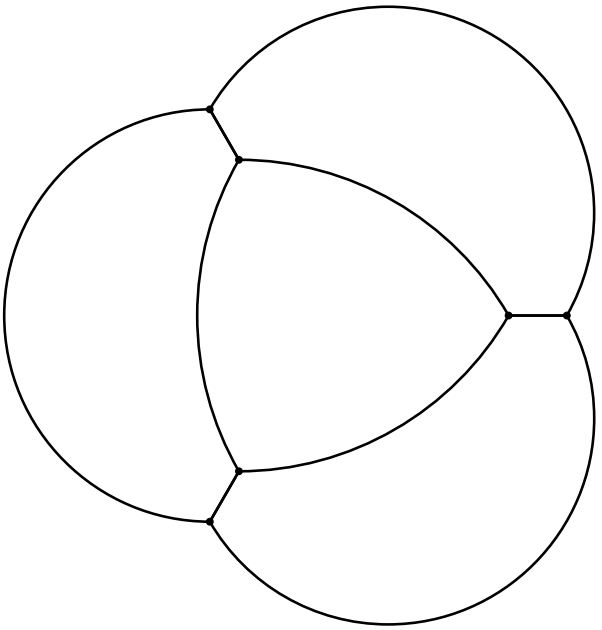


Figure 13: A symmetric four-bubble cluster

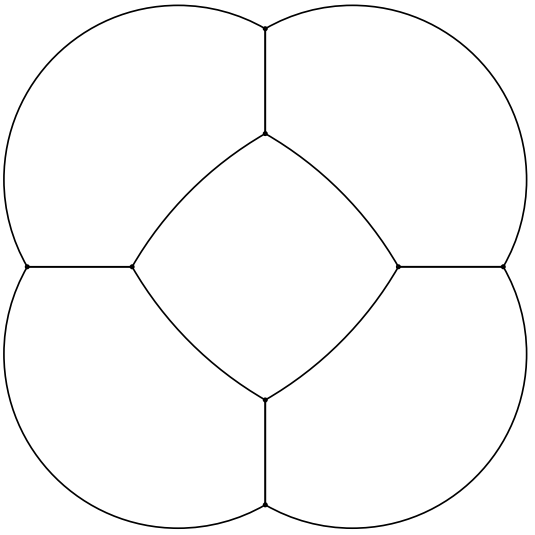


Figure 14: A symmetric five-bubble cluster

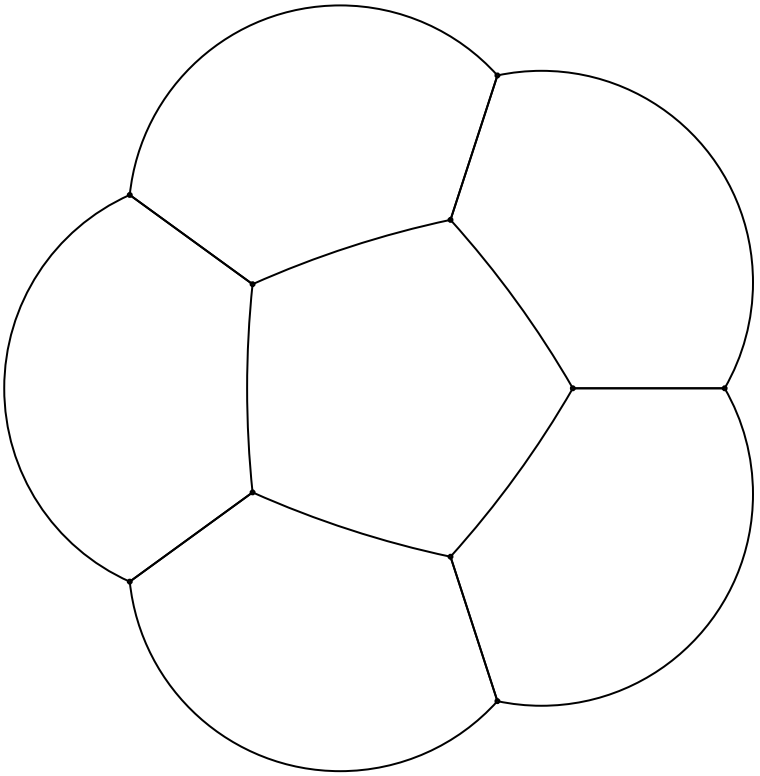


Figure 15: A symmetric six-bubble cluster

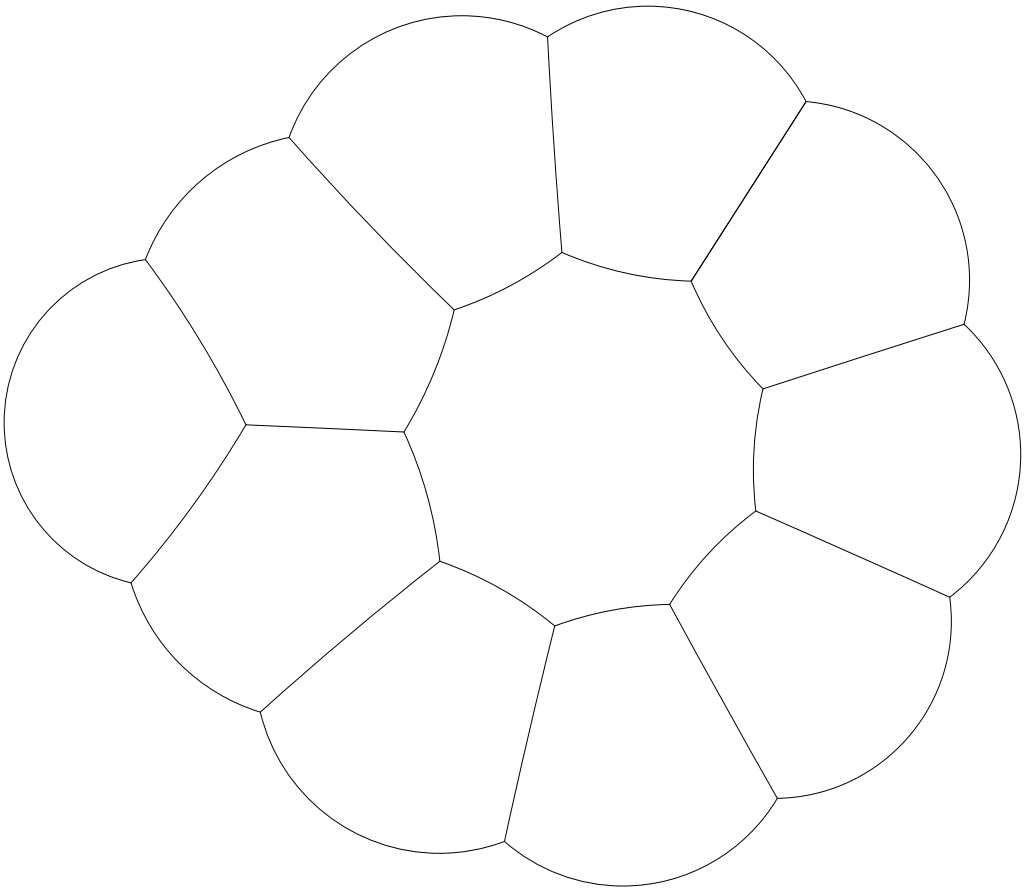


Figure 16: Reconstruction of an experimentally generated cluster

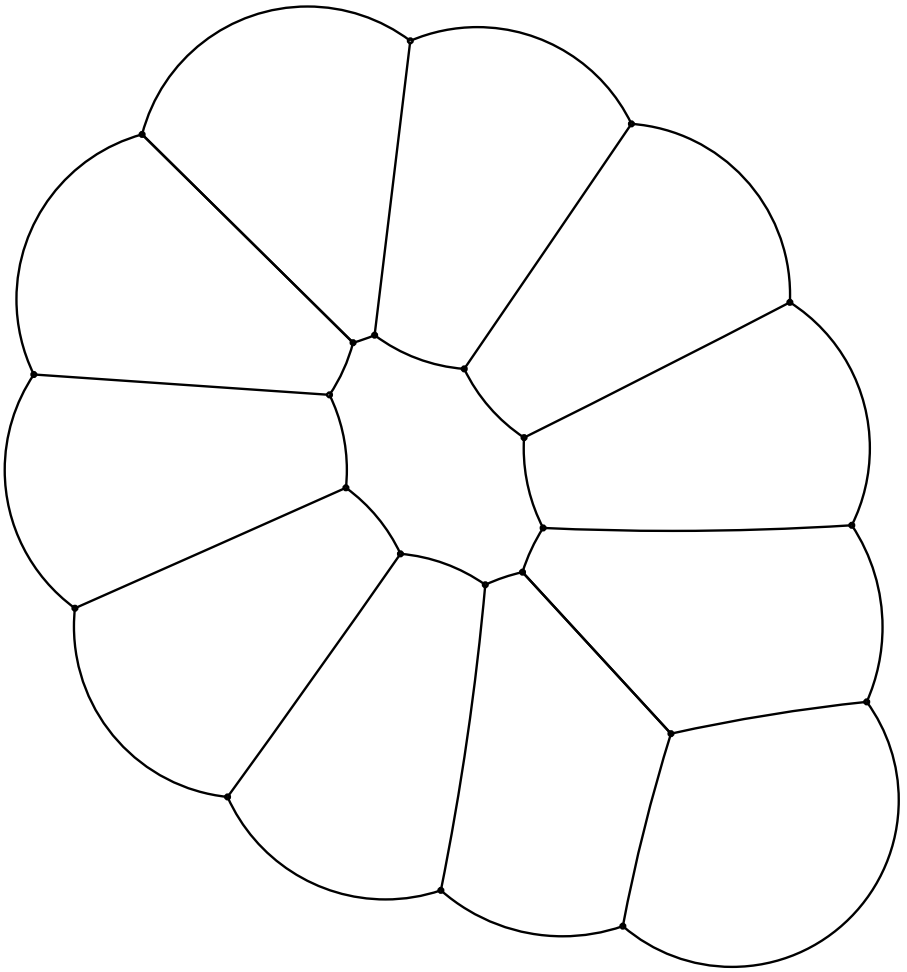


Figure 17: Recreation of a twelve bubble cluster generated by *Surface Evolver*. This figure may not be at equilibrium as the topology was not allowed to change while it was being created

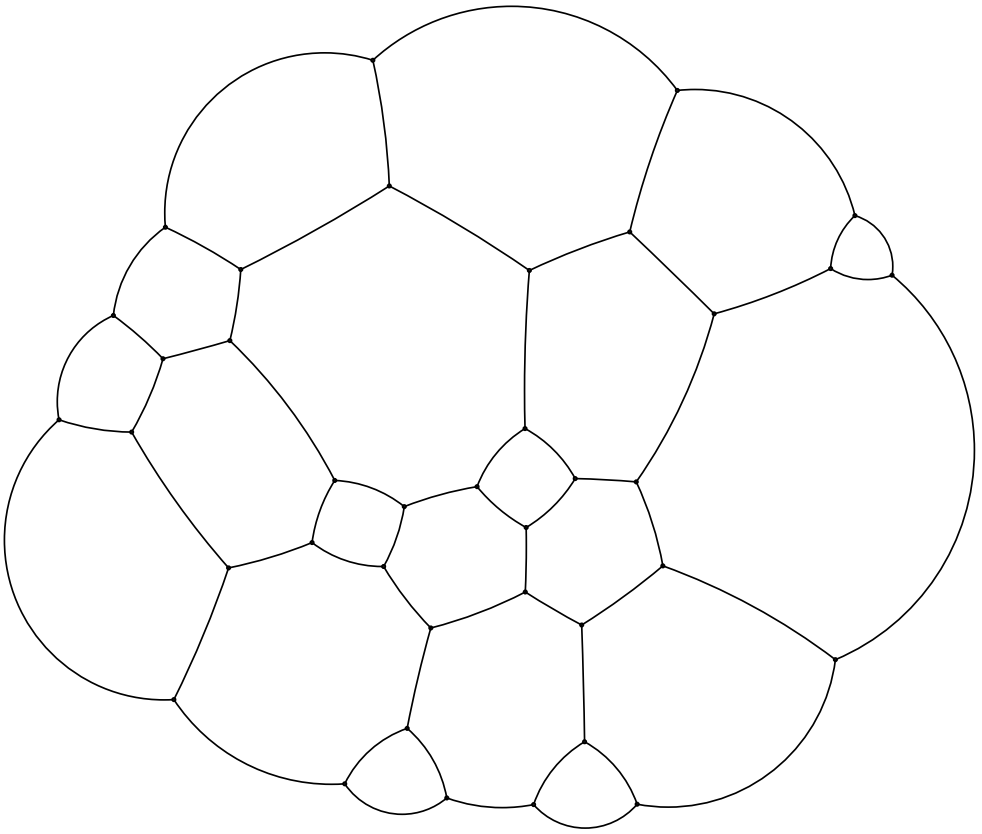


Figure 18: A twenty bubble cluster

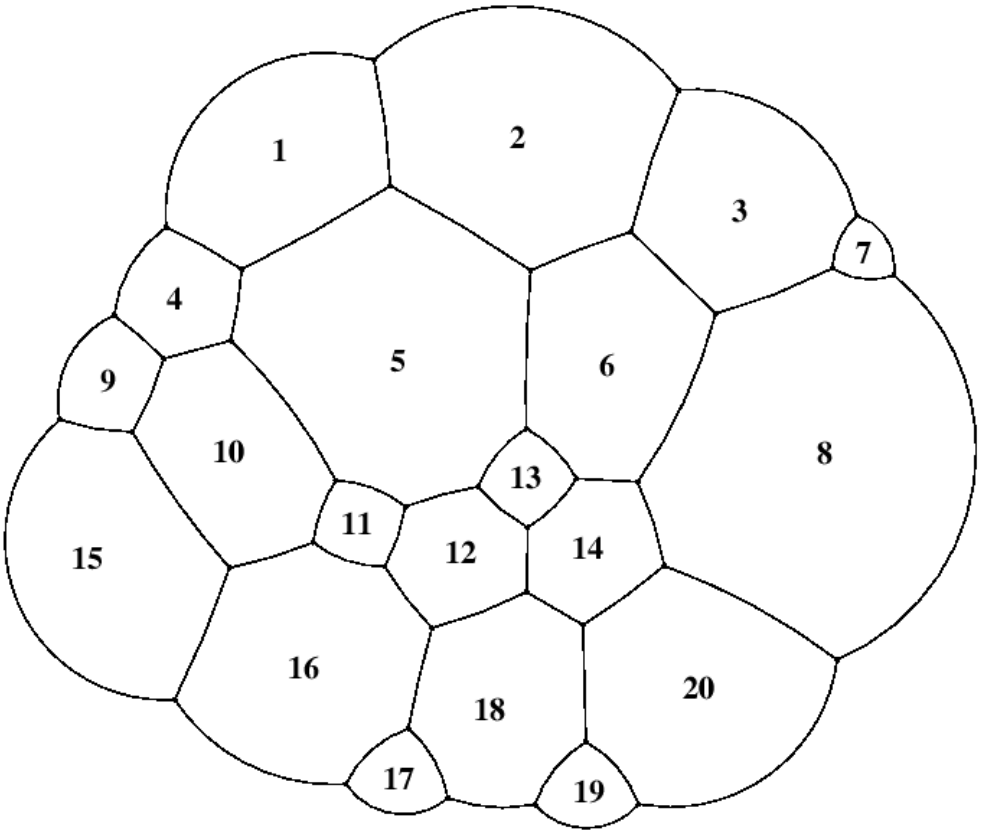



Figure 19: A twenty bubble cluster, including bubble indices

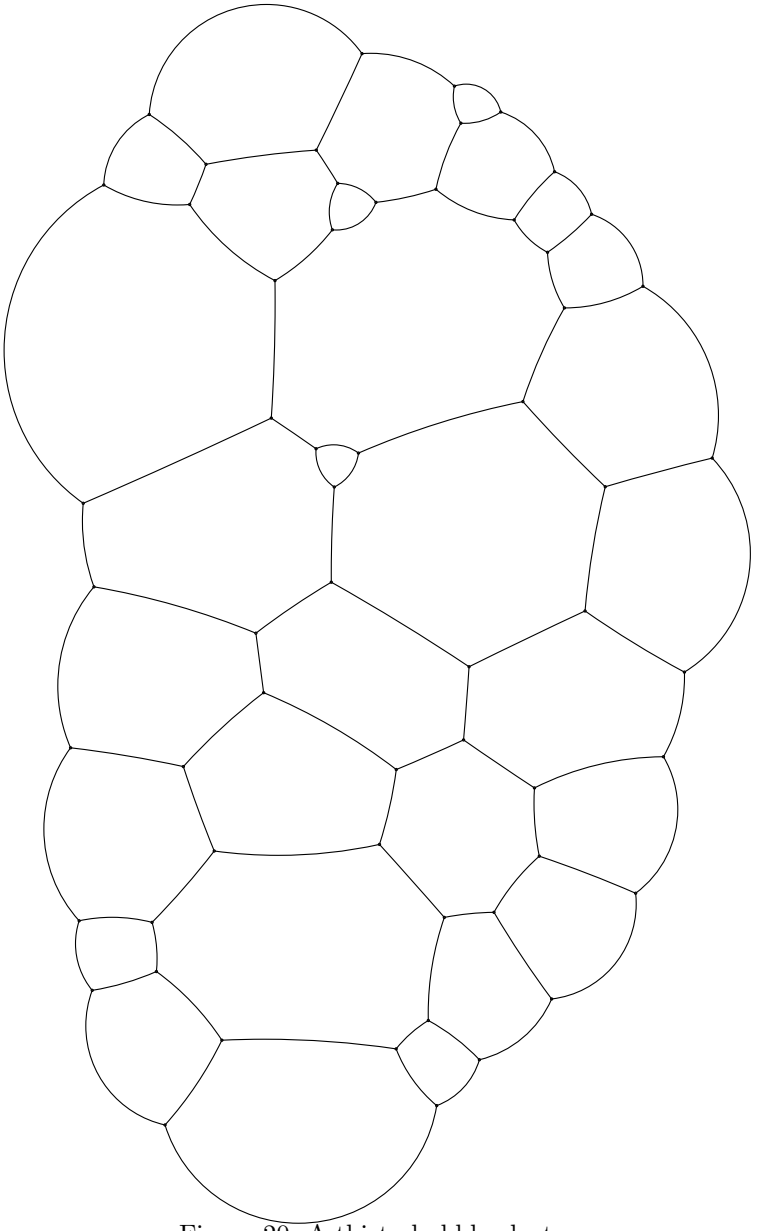


Figure 20: A thirty bubble cluster

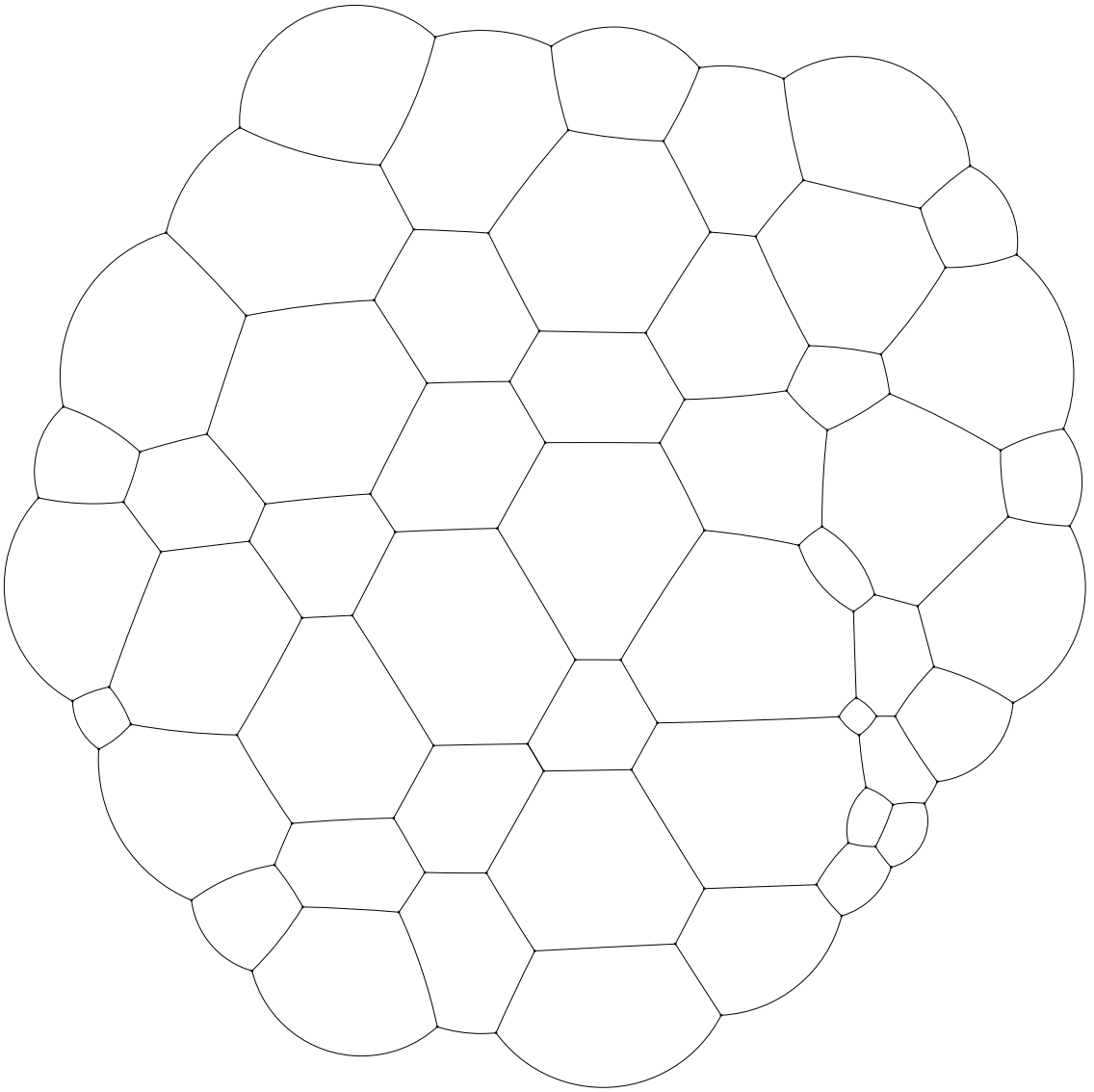


Figure 21: A fifty bubble cluster

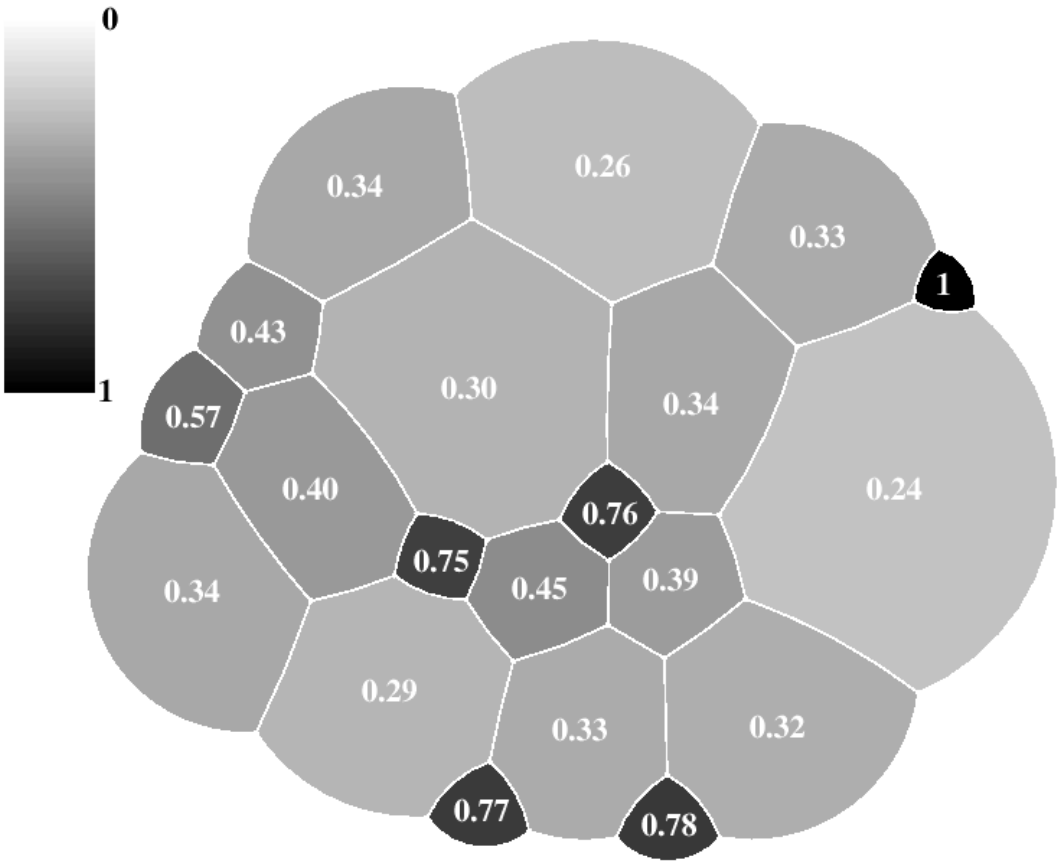



Figure 22: Normalised pressures of a twenty bubble cluster

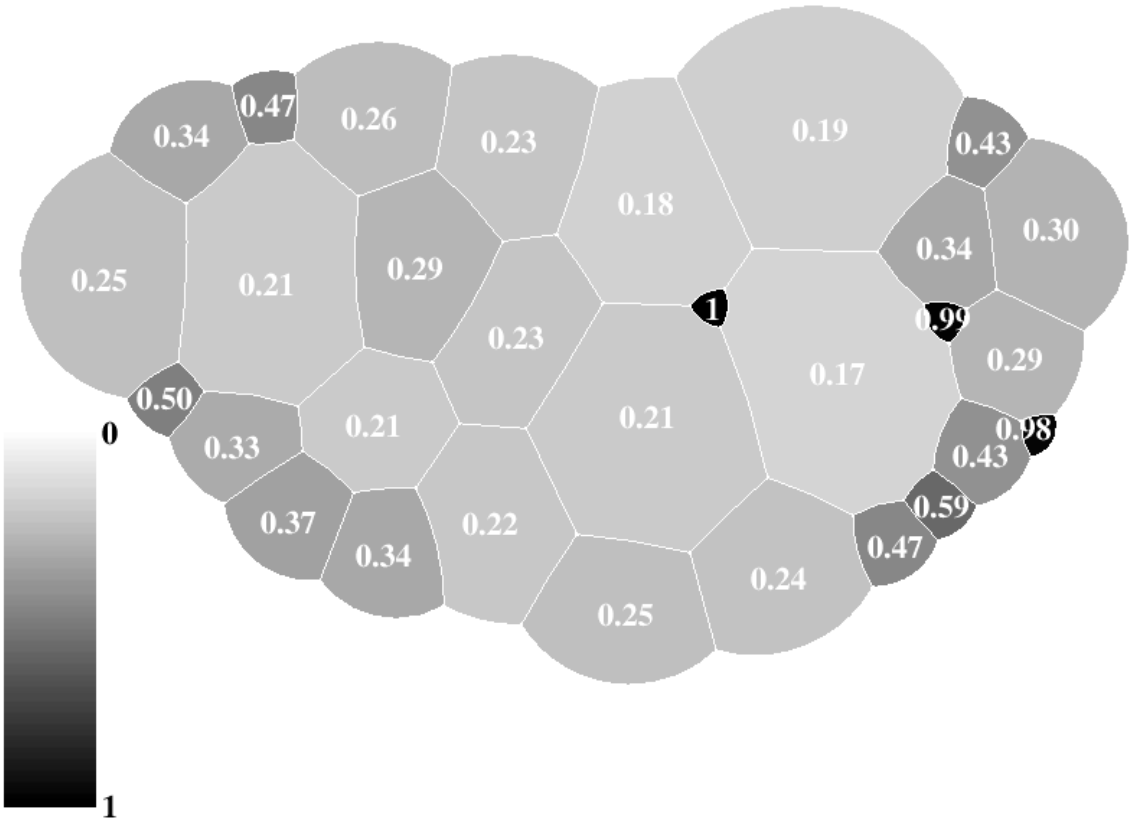



Figure 23: Normalised pressures of a thirty bubble cluster

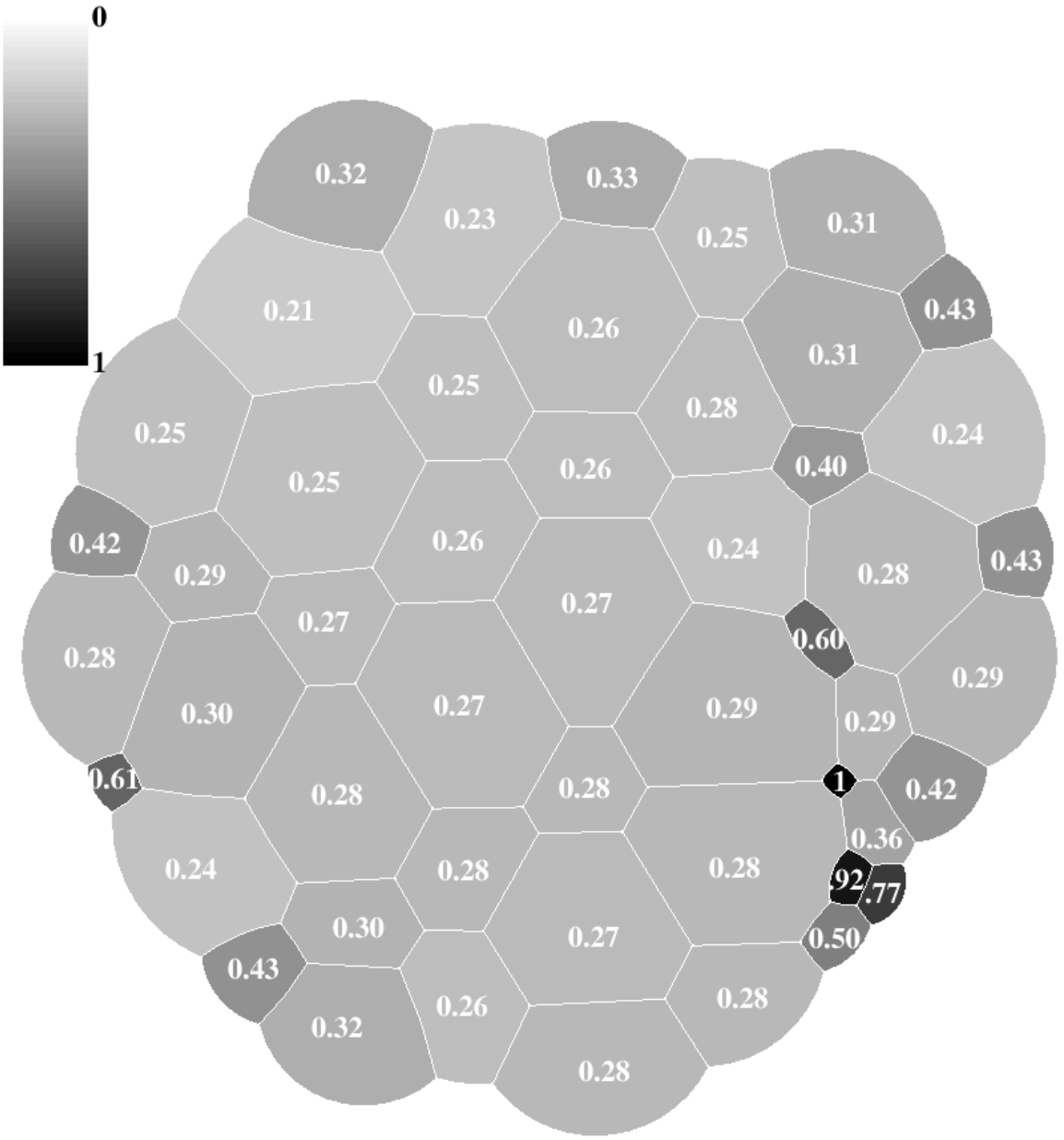



Figure 24: Normalised pressures of a fifty bubble cluster

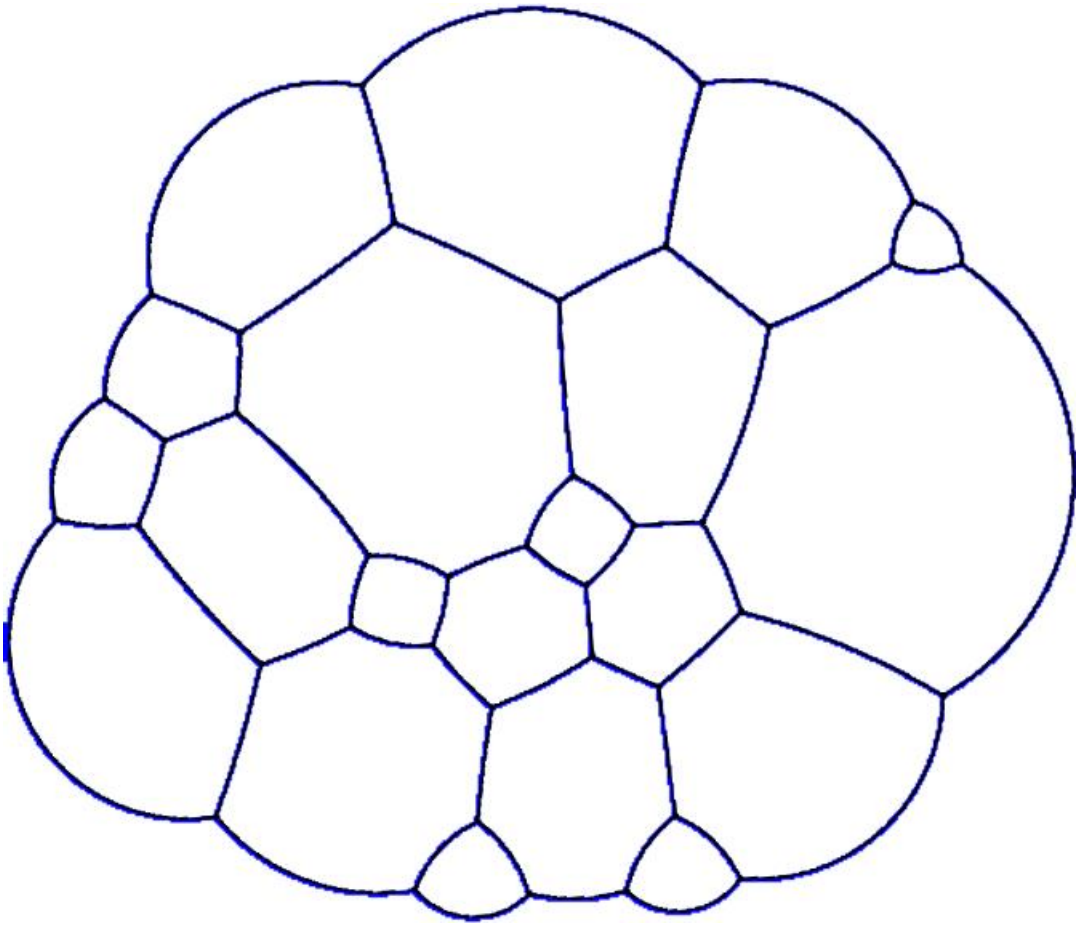


Figure 25: A reconstruction (in black) superimposed on *Surface Evolver's* creation (in blue)

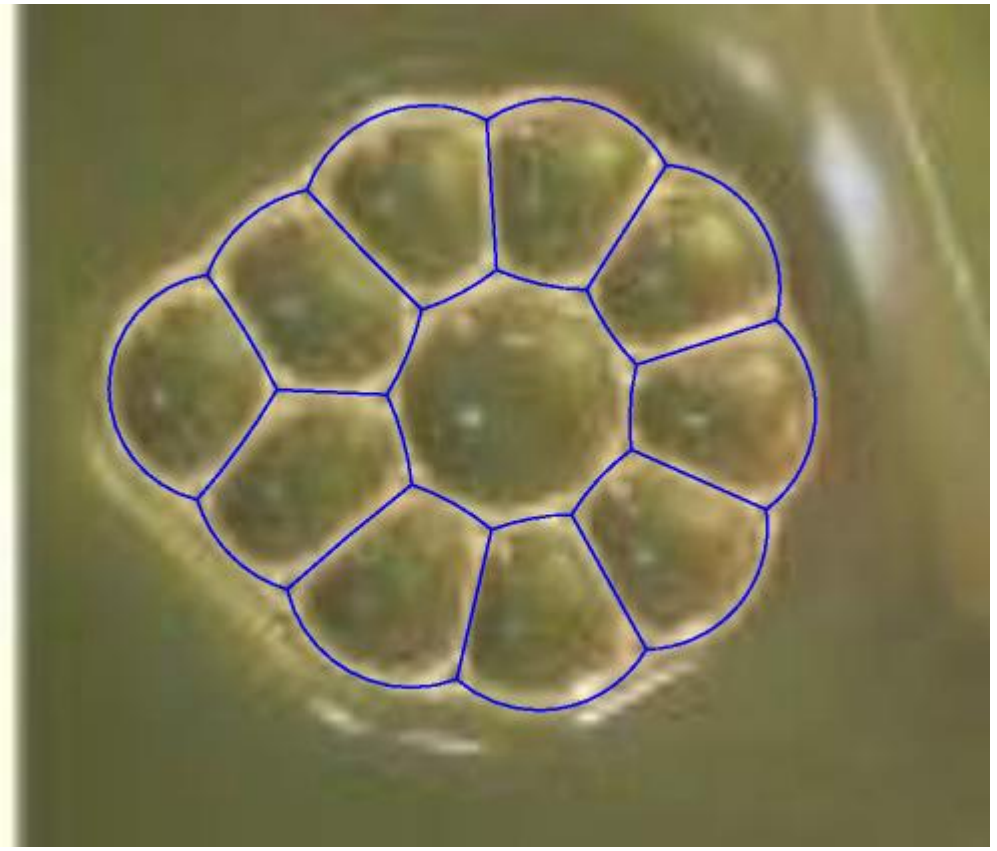


Figure 26: A reconstruction (in blue) superimposed on an experimentally generated image

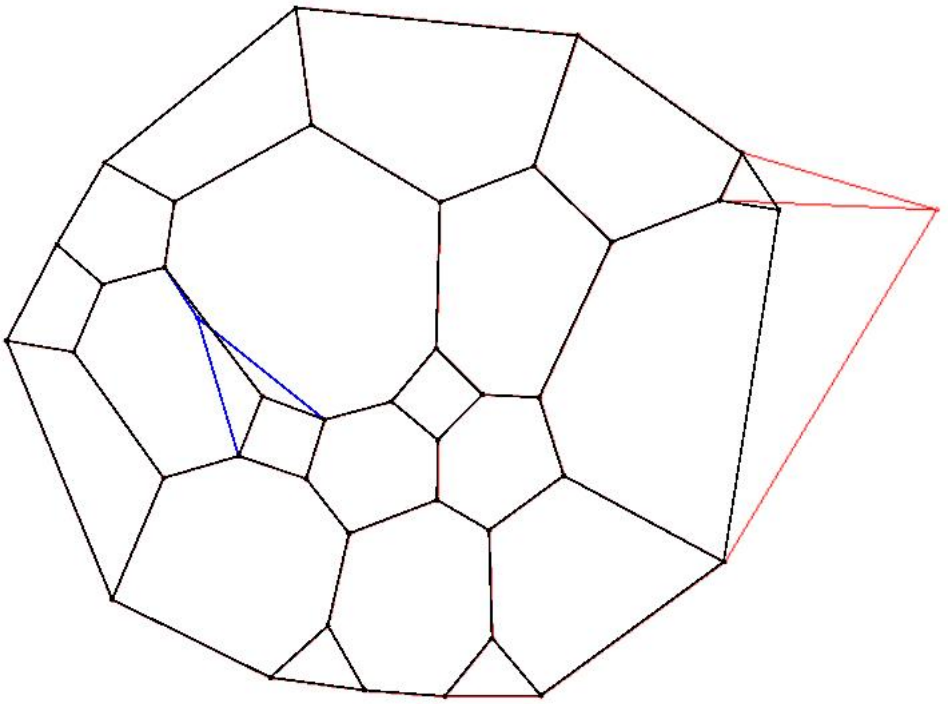


Figure 27: Inital conditions for non-equilibrium conditions. The black lines represent the equilibrium conditions, the red lines show a point on the outside being displaced and the blue lines show a point on the interior being displaced. One point has been moved in each case.

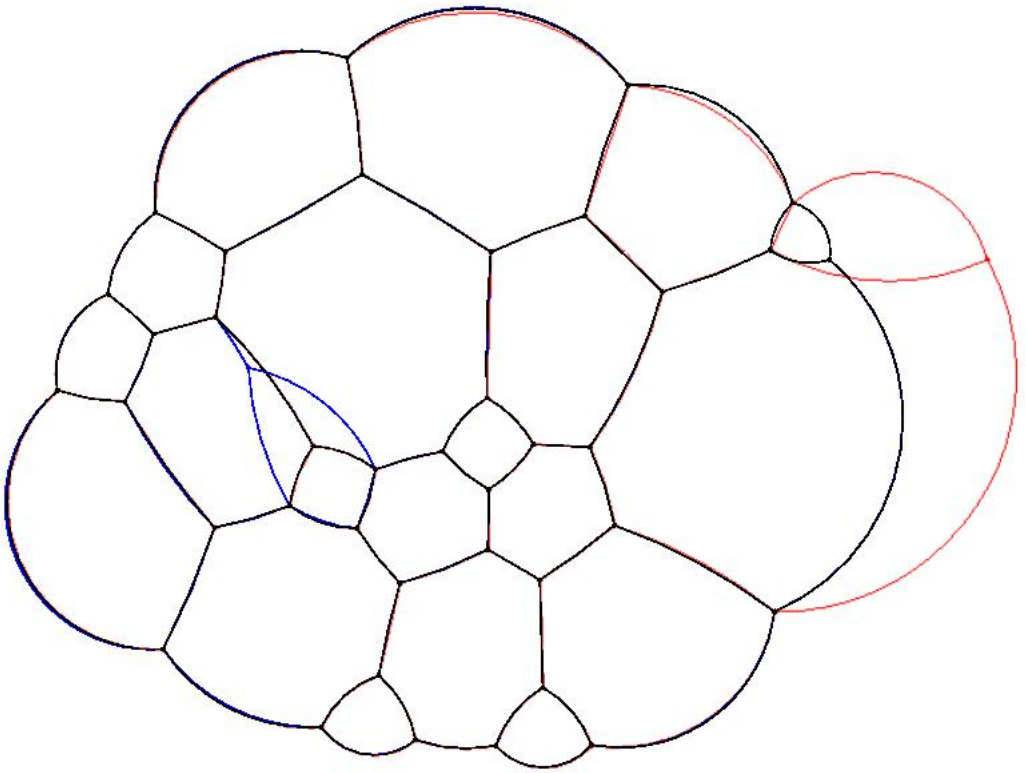


Figure 28: Reconstructed foams for non-equilibrium conditions. The black lines represent the equilibrium conditions, the red lines show a point on the outside being displaced and the blue lines show a point on the interior being displaced. It is apparent that the program will do its best to form a foam given the initial veritces, if those vertices do not represent an equilibrium foam then the program will do the best it can.