Relationships between areas in a triangulation of a square

By

Phillip Andreae

Adviser: Aaron Abrams

Department of Mathematics and Computer Science

Relationships between areas in a triangulation of a square

By

Phillip Andreae

Adviser : Aaron Abrams

An abstract of
A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Department of Mathematics and Computer Science

2010

Abstract

Relationships between areas in a triangulation of a square
By Phillip Andreae

For a triangulation of the unit square—that is, a tiling of the square by triangles—we consider the general problem of studying the relationships between the areas of these triangles. For a particular combinatorial arrangement of vertices and edges, by a dimension count we expect there to exist a relation that must be satisfied by the triangles, regardless of where the vertices are placed. By generalizing the notion of triangulating a square and applying some facts from algebraic geometry, we can prove that this relation is in fact a homogeneous polynomial equation that is an invariant of the combinatorial triangulation. Our focus is on calculating the degree of this polynomial for any arbitrary triangulation. We develop and implement an algorithm to compute this degree by inductively relating a triangulation to simpler "factor" triangulations and studying the relationship between the associated polynomials.

Relationships between areas in a triangulation of a square

By

Phillip Andreae

Adviser: Aaron Abrams

A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Department of Mathematics and Computer Science

2010

Contents

List of Figures

1 Introduction

Fundamentally, this paper deals with the division of a square into triangles. We call such a division a triangulation and require that it obey some simple, intuitive rules. The basic problem is to study the areas of these triangles and the relationships between them.

1.1 A simple illustrative problem

To motivate this work and capture the attention of the cynical reader, we begin with a simple problem that illustrates the general question we want to ask. Consider the triangulation of a square represented in Figure 1—in that picture, a single point is placed inside the unit square and connected to each of the four corners to form four triangles that tile the square, labeled A, B, C, and D. If we wanted to, presumably we could calculate the area of each triangle in terms of the coordinates (x, y) of the point in the middle. The question we want to ask—and this is the question we want to ask for more complicated triangulations throughout this paper—is whether there is some relationship between A, B, C, and D that must always hold, no matter where (x, y) lies. Certainly we could not choose any four areas at random and expect some choice of (x, y) to make the four triangles have those areas. There must be some restriction on the four areas—can you see what it is?

Well, there are actually two restrictions, one of which may be more apparent than the other. The first is that since the triangles tile the unit square, the sum of the four areas must be equal to the area of the square. We obtain the following simple equation:

$$A + B + C + D = 1.$$

The second restriction is somewhat less obvious but still easy to understand, and if we actually write down each area in terms of x and y, it becomes quite apparent.

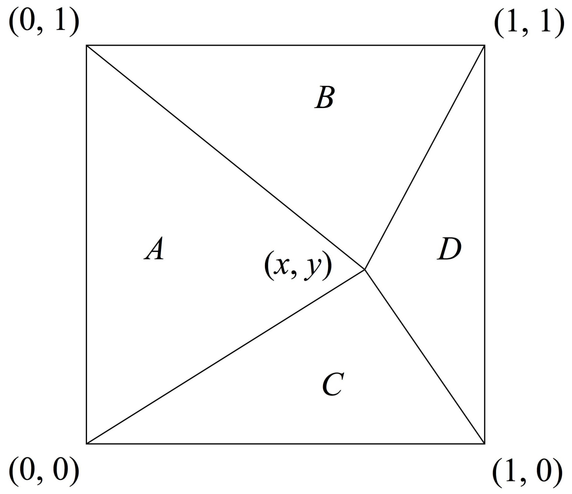


Figure 1: A simple triangulation of the unit square with one interior vertex and four triangles.

It is that each of the opposite pairs A and D, and B and C, must sum to half the area of the square. We can condense this information into another simple equation: $A + D = B + C = \frac{1}{2}$, or

$$A + D - B - C = 0.$$

Thus there are two equations that the four areas must always satisfy. For more complicated triangulations of the square with more triangles, we might expect something similar to happen. An analogue of the first equation, which states that the sum of the triangles' areas is equal to the area of the square, must always hold, of course. The second equation, though, which is specific to the triangulation in Figure 1, is more interesting, and it is not immediately obvious whether there is always an analogous equation, although we might expect there to be. It turns out that we can in fact prove the existence of such an analogous equation for any triangulation of the square, but in general, this equation is much more difficult to find than for our simple example. The focus of this paper is precisely this equation.

1.2 Introduction to the remainder of the paper

We will now provide a summary of the content of this paper. First we consider a triangulation in an abstract, combinatorial sense as a set of rules for connecting a number of points or vertices by edges; these connections define triples of points that form triangles, which when put together, make up an abstract square. We then want essentially to place the triangulation in the plane by giving coordinates to the vertices, turning the edges into line segments, and turning the abstract triangles into real triangles, each of which has an area determined by the coordinates of its vertices. Together the triangles tile an actual square in the plane, with four "boundary" vertices making up the corners of the square. From an abstract combinatorial structure, we now have a geometric picture of a triangulation of a square, just like the picture in Figure 1. In fact, though, we can create an infinite number of pictures for any triangulation depending on where exactly the vertices are placed. The key step is in recognizing that no matter how we draw the picture, the areas of the resulting triangles must maintain certain relationships to each other in ways that are dependent only on the combinatorial triangulation. Our goal, put simply, is to study these relationships. In particular, we study a certain polynomial relation that is always satisfied by the areas.

To do this rigorously, we must generalize things a bit, allowing the picture of a triangulation to exist not just in the Euclidean plane $\mathbb{R}^2$, but in the bigger space $\mathbb{C}^2$. Basic principles of algebraic geometry, which is in part the study of polynomial equations, are quite useful in studying our polynomial relation. Unfortunately, in return for more powerful mathematical tools, some of our intuition must be sacrificed. Much intuition still remains, though, and we can still generally imagine simple pictures of triangles inside a square in the plane, although the underlying mathematics is somewhat more complicated. We will attempt to emphasize the use of this intuition throughout this paper.

As mentioned, our primary interest is in a polynomial whose “variables” are the areas of triangles associated to a triangulation. This polynomial, though, is often quite complicated and can contain a large number of terms, which are hard to pin down in general. Specifically, then, we will attempt to study the degree of the polynomial, which is just a single integer that is a well-defined function of the combinatorial triangulation alone. To study the polynomial and its degree for a general complex triangulation, we apply an inductive algorithm to reduce it to a related but simpler triangulation. The essential step in the algorithm is asking the following question: given a triangulation, what would happen if a particular triangle weren't there (meaning, really, if that triangle had area zero)? We refer to the process of setting the areas of certain triangles equal to zero as “killing” triangles. Killing a triangle to produce a new triangulation has implications both for the possible pictures of the triangulation and for the associated algebra, but those implications are linked. Furthermore, the polynomial associated to the new triangulation is related to the one associated to the original triangulation in a predictable way. The algorithm proceeds inductively by killing series of triangles until triangulations whose polynomials have known degree are reached.

This algorithm provides a theoretical method for computing the degree of the polynomial for a triangulation, and we think we are close to proving that it works for any triangulation. Proving the existence of such a foolproof technique is an end in itself, but we are also interested in classifying triangulations and making general statements about the degrees of the polynomials associated to different triangulations. All triangulations with linear (degree one) polynomials have been classified, for example, but little is known about what kinds of triangulations give quadratic polynomials. We hope that this algorithm will serve as a computational tool in answering questions like these.

It is worth noting that the algorithm is actually quite simple and can theoretically be implemented by hand. For reasonably complicated triangulations, however, it can become quite long and tedious, and implementation is much easier with the help of a computer. We have used Java to program an application to implement the algorithm, although its development is an ongoing process. In the future we hope to use this program to help us classify triangulations and their polynomials.

This paper begins with an introduction to the theoretical framework we use in studying triangulations. In the context of this framework, we then proceed to prove the existence of the polynomial in question and discuss the theory behind our algorithm for computing this polynomial's degree, which first requires an introduction to some principles of algebraic geometry. Finally, we discuss the implementation of this algorithm and some of the associated practical issues.

2 Preliminaries

2.1 Honest triangulations

The basic object of study for us is a triangulation. We always want to keep in mind a tangible picture of a triangulation as a tiling of a square in the Euclidean plane by triangles, but more fundamental in a sense is the abstract connectedness of the vertices. Here we introduce triangulations in a general combinatorial sense.

A triangulation is a simple enough topological and combinatorial object; it can be thought of as a planar graph containing a number of vertices, a number of edges consisting of two distinct vertices, and a number of faces, which for a triangulation are subject to certain restrictions. We are specifically interested in triangulations of a disk with exactly four vertices and four edges on the boundary; these are to become the boundary of the square. In order to be considered a triangulation, each face inside the disk must in fact be a triangle, meaning it is bounded by exactly three edges and three vertices, which are connected pairwise by the three edges. Furthermore, every edge is shared by exactly two triangles, with the exception of the four boundary edges, which each appear in only exactly one triangle. For every triangulation, this definition will allow us to draw a unique class of pictures in the plane representing an actual division of a square into triangles.

Let $\mathcal{T}$ be a triangulation satisfying these requirements with k interior vertices. Denote the four boundary vertices by $0, 1, 2,$ and 3 and the interior vertices by $v_1, \dots, v_k$. The set of all vertices $\{0, 1, 2, 3, v_1, \dots, v_k\}$ is denoted by $V(\mathcal{T})$ and contains $k+4$ elements. The triangulation is specified by $V(\mathcal{T})$ along with a set of m edges, where each edge is an unordered pair of two vertices $\{w_1, w_2\}$, with $w_1, w_2 \in V(\mathcal{T})$, and by a set of n triangles, where each triangle is an ordered triple of vertices (w_1, w_2, w_3), or, equivalently, an ordered triple of edges (e_1, e_2, e_3), where $e_1 = \{w_1, w_2\}$, and so on.

By ordering the vertices of each triangle, we are giving an orientation to the triangulation. We specify that only the relative order of the vertices matters, so that are only two distinct ways to order a triangle; that is, for the triangle with vertices $w_1, w_2,$ and w_3, the order (w_1, w_2, w_3) is equivalent to (w_3, w_1, w_2) and (w_2, w_3, w_1), but all of these are distinct from (w_1, w_3, w_2), (w_2, w_1, w_3), and (w_3, w_2, w_1). We require that the orientation of the triangulation be consistent in the following sense: if triangles T and T' share the vertices w_1 and w_2, with their third vertices being u and v, respectively, then the orientations can be written $T = (w_1, w_2, u)$ and $T' = (w_2, w_1, v)$. We can think of the edge $\{w_1, w_2\}$ as having a different direction in T and T'.

Note that this consistency requirement still allows for exactly two distinct ways in which to orient the triangulation. To remove this ambiguity, we specify by convention that the triangle containing the edge $\{0, 1\}$ (this triangle exists and is unique) have a certain orientation: if u is the third vertex, the triangle has an orientation equivalent to $(0, 1, u)$. Specifying the orientation of this triangle then completely determines the orientation of every other triangle according to the consistency rule.

The boundary vertices are numbered 0, 1, 2, and 3 for a reason: this is in a sense their order around the boundary of the disk, or around the boundary of the square. This means that we require that the four boundary edges be $\{0,1\}$, $\{1,2\}$, $\{2,3\}$, and $\{3,0\}$. If we like, in a picture of a triangulation, we can think of the numbers 0-3 as numbering the corners of the square going counterclockwise. As a convention, we start the numbering at 0 to represent that 0 is the "origin" in the plane, but this doesn't have to be the case. Following this convention, we will often refer colloquially to $\{0,1\}, \{1,2\}, \{2,3\},$ and $\{3,0\}$ as the bottom, right, top, and left edges, respectively.

Now we can interpret the orientation of a triangle in another way: it essentially specifies the direction of a small clockwise- or counterclockwise-oriented circle inside that triangle in a picture of the triangulation. By specifying the orientation $(0,1,u)$, we are really requiring that every one of these circles is oriented counterclockwise, if we think of the boundary vertices as numbering the corners of the square in counterclockwise order. This will give us a consistent definition of the area of a triangle later.

We call a triangulation defined in this way an honest triangulation. The next step will be to add to this definition to create a more general idea of a triangulation, but an honest triangulation, for which a simple picture exists, will always be the starting point.

2.2 General triangulations

For our purposes, a triangulation will really be specified by an honest triangulation $\mathcal{T}$ as described above along with a set of "conditions" $\mathcal{C}$ and an equivalence relation $\sim$ such that:

1. $\mathcal{C}$ is a collection of conditions where each condition is a union of triangles that forms a connected subgraph of the dual graph of $\mathcal{T}$ on a torus

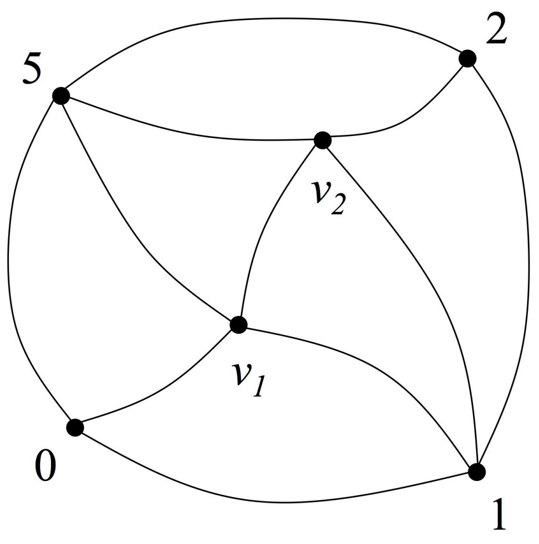


Figure 2: A simple “honest” triangulation with two interior vertices and six triangles.

2. $\sim$ is an equivalence relation on the boundary vertices of $\mathcal{T}$ (0, 1, 2, and 3) satisfying the following:

   (a) $0 \sim 1 \Leftrightarrow 2 \sim 3$

   (b) $0 \sim 3 \Leftrightarrow 1 \sim 2$

   (c) $0 \sim 2$ or $1 \sim 3 \Rightarrow 0 \sim 1 \sim 2 \sim 3$

These definitions are quite abstract, so let’s explain what they mean. The set of conditions $\mathcal{C}$ is just a collection of sets of triangles. Each set of triangles must be “connected;” this can be defined rigorously in terms of the dual graph as above, but it really just means that in a picture of the honest triangulation, each set consists of a number of triangles whose union is a connected subset of the plane, where two triangles are defined to be connected if they share an edge (sharing a single vertex doesn’t count). The added condition that the dual graph is placed on a torus means that we identify the opposite boundary edges; that is, two triangles are also “connected” if one contains the boundary edge $\{0, 1\}$ and the other contains $\{2, 3\}$, or if one contains $\{1, 2\}$ and the other $\{0, 3\}$.

For each condition, we can also keep track of all the vertices that are contained in one of the triangles in that condition. This list of vertices, in fact, is more useful, and we will usually represent each condition by this list of vertices rather than by the original list of triangles. Later, we will see that the conditions actually represent sets of triangles that have zero area (have been "killed") and that the corresponding set of vertices represent vertices that must be collinear. For now, we leave the conditions as an abstract list of sets of triangles.

The equivalence relation $\sim$ is just a way to keep track of which boundary vertices must be "equal." Being equivalent with respect to $\sim$ will later correspond to having the same coordinates in a picture, and, as with the conditions, $\sim$ will be related to triangles having zero area. This will be explained in more detail.

The triple $(\mathcal{T}, \mathcal{C}, \sim)$ is a new object containing more information than the honest triangulation $\mathcal{T}$ alone, but for convenience, we will call such a triple a "triangulation" and usually refer to it simply as $\mathcal{T}$. We must remember that this is an abuse of notation and a "triangulation" generally consists of not just an honest triangulation but also a set of conditions and an equivalence relation.

2.3 Euler's formula

We now derive an important relationship between the number of interior vertices k and the number of triangles n of a triangulation $\mathcal{T}$ using Euler's formula. One simple statement of Euler's formula is the following: for a planar graph containing V vertices, E edges, and F faces, it is always true that

$$V - E + F = 2.$$

$\mathcal{T}$ already represents a planar graph, but it is more convenient to construct another planar graph $\mathcal{P}$ from $\mathcal{T}$ by adding a vertex "outside the disk" that is connected to each of the four boundary vertices by an edge. Figure 3 illustrates this construction. Note that $\mathcal{P}$ contains one extra vertex, four extra edges, and three extra faces relative to $\mathcal{T}$ (if we count the outer face in both graphs). This construction is convenient because the outer face of $\mathcal{P}$ is a triangle, whereas the outer face of $\mathcal{T}$ was a square. Thus every face of $\mathcal{P}$ contains exactly three edges and every edge is shared by exactly two faces, with no exceptions. This implies that for $\mathcal{P}$,

$$E = \frac{3}{2}F.$$

Euler's formula then states

$$V - \frac{1}{2}F = 2,$$

or, equivalently,

$$F = 2V - 4.$$

Since $\mathcal{P}$ was constructed from $\mathcal{T}$ by adding four faces and one vertex, we can count the number of faces and vertices of $\mathcal{P}$ in terms of the number of faces n and the number of interior vertices k of $\mathcal{T}$ in the following way: $F = n + 4$ and $V = k + 5$. Substituting these expressions, we obtain the following result:

$$n = 2k + 2.$$

The importance of this result will be seen shortly.

2.4 A naive discussion of relationships between areas

Up to this point we have discussed triangulations primarily as combinatorial objects; that is, as a set of vertices and a set of simple combinatorial relationships between them (the edges, triangles, conditions, and equivalence relation). Along the way, we have briefly mentioned the idea of a triangulation as a picture of a square and

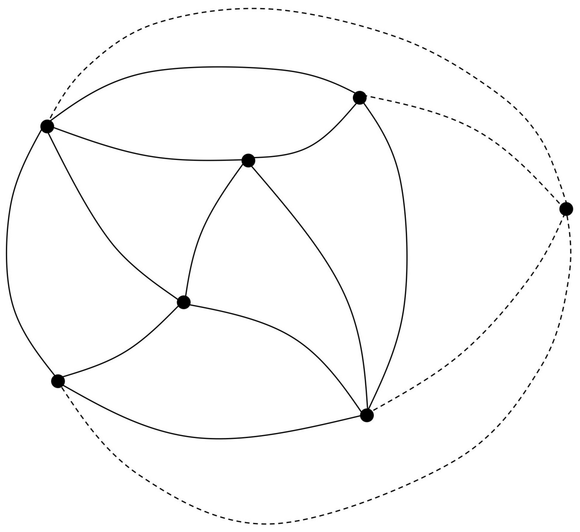


Figure 3: From a triangulation of the disk $\mathcal{T}$, we can construct a new planar graph $\mathcal{P}$ whose faces are all triangles by adding an extra vertex outside the disk.

triangles in the plane. Turning an abstract triangulation into this sort of concrete geometric object is quite natural, and this will be our next step. The discussion that follows treats this “concretization” in a naive but intuitively satisfying way.

Let $\mathcal{T}$ be an honest triangulation (meaning $\mathcal{C}$ is empty) with k interior vertices. We wish to “realize” $\mathcal{T}$, which is a triangulation of an abstract disk, as a triangulation of a real square in the plane. To do this, we assign to each of the vertices of $\mathcal{T}$ coordinates in $\mathbb{R}^2$, specifying that $0 \mapsto (0,0), 1 \mapsto (0,1), 2 \mapsto (1,1), 3 \mapsto (0,1),$ and $v_i \mapsto (x_i, y_i)$ for an interior vertex v_i, with $(x_i, y_i) \in [0,1] \times [0,1]$. Thus the four boundary vertices represent the corners, and the four boundary edges the sides, of the unit square, and the k interior vertices represents points inside the square. The abstract combinatorial triangles are now real geometric triangles, $2k+2$ in number, and the edges are the edges of these triangles; together the triangles tile the unit square. Figure 4 illustrates this assignment of coordinates in the plane to the vertices of an abstract triangulation

We may now speak of the areas of these triangles, which are easy to compute. The area of an oriented triangle T_{abc} defined by the ordered vertices (w_a, w_b, w_c)

whose coordinates are $(x_a, y_a), (x_b, y_b),$ and (x_c, y_c), respectively, is calculated via a determinant of a matrix containing the coordinates:

$$A_{abc} = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ x_a & x_b & x_c \\ y_a & y_b & y_c \end{vmatrix}.$$

Note that the columns of this matrix are arranged in the same order as the vertices of the triangle. Here is where the consistency of the orientation of the triangulation comes in: in order to make the areas well-defined, we must essentially specify in what order to place the columns. If we had changed the orientation of the triangle, this would correspond to interchanging two of the columns of the matrix, producing the negative of the original determinant and the negative of the original area. The above choice of order corresponds to a counterclockwise orientation of the vertices of the triangle, which is consistent with our convention.

Having defined the area of a triangle given its coordinates, we can define a map from the set of coordinates of the interior vertices to the areas of the triangles. These coordinates are $2k$ in number (two for each of k vertices), and the triangles are $2k+2$ in number, so we have a map $f\colon \mathbb{R}^{2k} \to \mathbb{R}^{2k+2}$, where each coordinate function of f is a polynomial of degree at most two. Continuing somewhat naively, if we assume that f is locally one-to-one (a reasonable assumption), a theorem of algebraic geometry gives that the image of f is isomorphic to $\mathbb{R}^{2k}$. This image, then, has codimension two in the codomain, so we might intuitively expect that the areas $A_1, \ldots, A_{2k+2}$ satisfy two relations, each of which corresponds to a difference in dimension of 1 between the codomain and the image of f. Another theorem of algebraic geometry, in fact, assures us of this.

It turns out that one of these relations is obvious: since the triangles tile the unit square, the sum of the areas is equal to the total area of the square, which is just 1:

$$\sum_{i=1}^{2k} A_i = 1.$$

The other relation is not so obvious, but its existence seems clear by the naive argument above. This relation will be the focus of the remainder of this paper; it turns out that it is in fact a polynomial relation, and by generalizing the idea of a realization and doing some technical work, we will show that it is a homogeneous polynomial. Our eventual goal will be to calculate the degree of this polynomial, which is uniquely determined by the combinatorics of the original triangulation $\mathcal{T}$.

To do all this, though, we will need some basic facts from algebraic geometry, and in particular we must become familiar with projective space. To this end, we now take a short detour to introduce projective space and some other relevant ideas about polynomials and projective varieties.

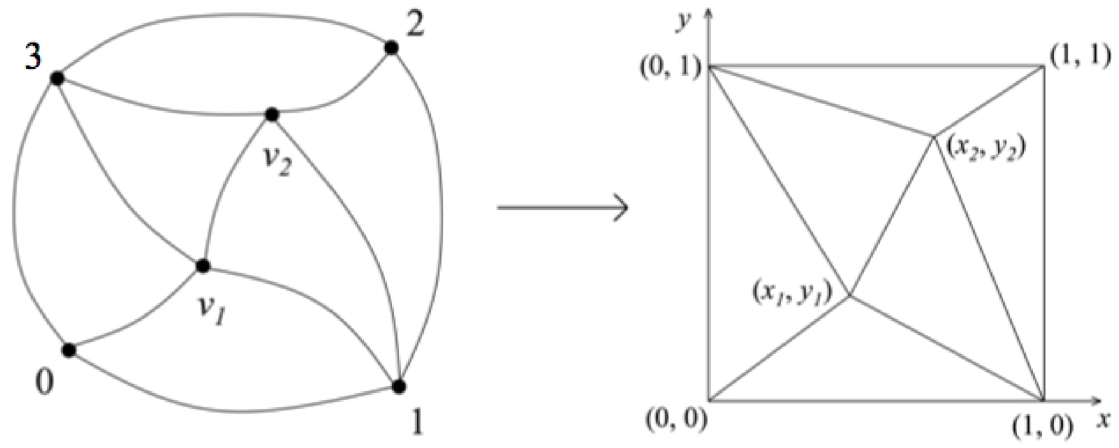


Figure 4: Turning an abstract triangulation of a disk into a tiling of the unit square by assigning coordinates to the vertices.

3 A brief introduction to algebraic geometry

Algebraic geometry is essentially the study of zero sets of polynomials. Since we are interested in a particular polynomial associated to a triangulation, we might expect that we will find some principles of algebraic geometry to be relevant. This section introduces some of these principles and is organized into three subsections, each of which treats a particular object or concept in algebraic geometry. The material on projective space and projective varieties is primarily adapted from [5], which provides a good introduction to the subject. [2] was also consulted and is another good resource on algebraic geometry. For a further introduction to polynomial rings and related topics in algebra, the curious reader is advised to consult [3].

3.1 Polynomials

First, a brief review of polynomials is in order. We must begin with a field $\mathbb{F}$, which is essentially just a set of "numbers" on which the operations of addition, subtraction, multiplication, and division make sense. The set of all polynomials in $x_1, \ldots, x_n$ with coefficients in $\mathbb{F}$ is denoted $\mathbb{F}[x_1, \ldots, x_n]$; this set forms a ring, meaning we can add, subtract, and multiply (but not always divide) polynomials to obtain new polynomials in a natural way.

To every polynomial, we can associate a nonnegative integer called its degree. The degree of a monomial (a polynomial with only one term) is just the sum of its exponents, and the degree of a general polynomial is defined as the maximum degree of its monomials or terms.

An especially important type of polynomial for us is a homogeneous polynomial. A polynomial is defined to be homogeneous if every monomial has the same degree. For example, $x^5 - 3x^3y^2 + 2xy^4$ is a homogeneous polynomial in two variables of degree 5 because the sum of the exponents in every term is 5. The polynomial in three variables $xz+yz^3$, on the other hand, is not homogeneous because xz has degree 2 and yz^3 has degree 4. Homogeneous polynomials are interesting in part because they are "quasilinear" in the following sense: for a homogeneous polynomial p of degree $d \geq 1$, $p(\lambda x_1, \ldots, \lambda x_n) = \lambda^d p(x_1, \ldots, x_n)$ for any scalar $\lambda \in \mathbb{F}$. In particular, if $\lambda \neq 0$, $p(\lambda x_1, \ldots, \lambda x_n) = 0$ if and only if $p(x_1, \ldots, x_n) = 0$, a fact which will be important later.

The notion of irreducibility of polynomials is another significant one. A polynomial $p \in \mathbb{F}[x_1, \ldots, x_n]$ is defined to be reducible if it can be expressed as the product of two nonconstant polynomials: $p = q_1 q_2$, where $q_1, q_2 \in \mathbb{F}[x_1, \ldots, x_n]$ with $\deg q_1 \geq 1$ and $\deg q_2 \geq 1$. If p cannot be expressed in this way, it is said to be irreducible; this means that whenever p is written as $p = q_1 q_2$, either $q_1 \in \mathbb{F}$ or $q_2 \in \mathbb{F}$. It is a fact that every polynomial p can be factored into a product of irreducible factor polynomials: $p = q_1 \ldots q_r$, where each q_i is irreducible. This factorization, furthermore, is unique up to multiplication of each factor by elements of $\mathbb{F}$.

It is worth noting that in $\mathbb{C}[x]$, the ring of polynomials in one variable with complex coefficients, the fundamental theorem of algebra gives that every polynomial can be factored into linear factors; that is, every polynomial has a "root" or solution. For this reason, we say that the complex numbers $\mathbb{C}$ are algebraically closed. This makes them a particularly convenient field to work with; for this reason, we will soon focus our attention only on polynomials with complex coefficients.

3.2 Projective space

Now we move on to discuss some basic principles of algebraic geometry. First, we will introduce the concept of projective space. Again, we will be particularly interested in complex projective space, but for now we will keep things more general: for any field $\mathbb{F}$, $\mathbb{F}^k$ is defined as the set of of n-tuples of elements of $\mathbb{F}$. $\mathbb{F}^k$ is said to be a vector space over $\mathbb{F}$, meaning we can add two elements of $\mathbb{F}^k$ and multiply elements of $\mathbb{F}^k$ by a scalar in $\mathbb{F}$ linearly. Its dimension over $\mathbb{F}$ is k.

$\mathbb{F}^n$ is an example of an affine space—that is, a space that is in a sense analogous to Euclidean space $\mathbb{R}^n$. To define projective space, which is a new type of space obtained from an affine space, define an equivalence relation $\simeq$ on $\mathbb{F}^k \setminus \{(0,\dots,0)\}$ by

$$u \simeq v \Leftrightarrow u = \lambda v \text{ for some } \lambda \in \mathbb{F} \setminus \{0\}.$$

The projective space $\mathbb{FP}^n$ is defined simply as the set of equivalence classes of $\mathbb{F}^{n+1} \setminus \{(0,\dots,0)\}$ with respect to $\simeq$:

$$\mathbb{FP}^n = \left(\mathbb{F}^{n+1} \setminus \{(0,\dots,0)\}\right) \big/ \simeq .$$

This space can be thought of as the set of all lines through the origin in $\mathbb{F}^{n+1}$ since $\simeq$ effectively equates all points that lie on the same line through the origin (all points that are a scalar multiple of each other). $\mathbb{FP}^n$ can be thought of as a compactification of $\mathbb{F}^n$, meaning it is an extension of $\mathbb{F}^n$ to include extra points at infinity.

Whereas a point in $\mathbb{F}^{n+1}$ is usually represented by an $(n+1)$-tuple $(x_1,\dots,x_{n+1})$, a point in $\mathbb{FP}^n$ is usually represented by an expression of the form $[x_1 : \cdots : x_{n+1}]$. The colons, perhaps, emphasize that only the ratios of the coordinates are significant in projective space. Since projective space consists of a set of equivalence classes of $\mathbb{F}^{n+1}$, a point in projective space can generally be written in many different ways depending on which representative of the equivalence class is chosen.

We will usually be interested in complex projective space, denoted by $\mathbb{CP}^n$. As an instructive example to gain some intuition about projective space, though, it is worthwhile to study $\mathbb{RP}^2$, also called the real projective plane. $\mathbb{RP}^2$ consists of all lines through the origin in $\mathbb{R}^3$; thus the points $(1,1,1), (2,2,2),$ and $(-\pi,-\pi,-\pi)$ in $\mathbb{R}^3$ all represent the same point in $\mathbb{RP}^2$ since they all lie on the line through the origin and $(1,1,1)$ (note they can also be written as scalar multiples of each other). For any point $[x : y : z]$ in $\mathbb{RP}^2$ with $z \neq 0$, we can scale so that $z = 1$ and rename the point as $[x' : y' : 1]$, where $x' = \frac{x}{z}$ and $y' = \frac{y}{z}$. In doing so we are effectively taking the radial *projection* of the point onto the plane $z = 1$ and defining it by just two coordinates representing the x- and y-coordinates of that projection.

This can be done for all the points in $\mathbb{R}^3$ except those for which $z = 0$; that is, the points in the xy-plane. These points have no projection onto the plane $z = 1$ because the lines through the origin in the xy-plane never intersect the plane $z = 1$. They correspond to the set of directions or lines through the origin in the xy-plane, which is actually just $\mathbb{RP}^1$. $\mathbb{RP}^2$, then, can be thought of as the ordinary real plane with a copy of $\mathbb{RP}^1$, the projective line, added: $\mathbb{RP}^2 = \mathbb{R}^2 \cup \mathbb{RP}^1$.

This is actually a general fact that is true for a projective space over any field: $\mathbb{FP}^n = \mathbb{F}^n \cup \mathbb{FP}^{n-1}$. We can see, then, that the idea of the construction of projective space is to take the points in an affine space of dimension $n + 1$ and "project" them onto a hyperplane of dimension n to obtain a copy of $\mathbb{F}^n$. For some special points, though, this is not possible, and we must remember to adjoin those points, which together look like projective space of dimension $n - 1$. These extra points can be thought of as points at infinity in $\mathbb{F}^n$.

3.3 Projective varieties

From this point on, we will focus our attention on the field $\mathbb{C}$ and complex projective space. We want somehow to apply polynomials to projective space, and in particular we want to look at zero sets of polynomials in projective space. But because projective space is only defined up to scalar multiples, the zero set of a general polynomial is not well-defined. What is well-defined, though, is the zero set of a *homogeneous* polynomial, since $p(\lambda x_1, \ldots, \lambda x_k) = 0$ if and only if $p(x_1, \ldots, x_k) = 0$ for a homogeneous p and $\lambda \neq 0$. For a homogeneous $p \in \mathbb{C}[x_1, \ldots, x_{n+1}]$, we denote by $Z(p)$ the zero set of p in $\mathbb{CP}^n$:

$$Z(p) = \{[x_1 : \cdots : x_{n+1}] \in \mathbb{CP}^n \colon p(x_1, \ldots, x_{n+1}) = 0\}.$$

A projective variety V, which is a fundamental object of study in algebraic geometry, is a subset of projective space that can be written as the common zero set of a collection of polynomials p_i:

$$\begin{aligned} V &= \bigcap_{i \in I} Z(p_i) \\ &= \{[x_1 : \cdots : x_{n+1}] \in \mathbb{CP}^n \colon p_i(x_1, \ldots, x_{n+1}) = 0 \;\; \forall i \in I\} \end{aligned}$$

Note that the index set I can generally be infinite and even uncountable; for our purposes, though, it will always be finite. In this case, we usually denote a variety by $V = Z(p_1, \ldots, p_s)$ to indicate that it is the common zero set of the finite collection of polynomials $p_1, \ldots, p_s$. In particular, we will be especially interested in varieties that are defined by a single polynomial; that is, varieties of the simple form

$$V = Z(p),$$

where p is a homogeneous polynomial (of positive degree). These kinds of varieties are called hypersurfaces and are analogous to what we typically think of as surfaces in the affine space $\mathbb{R}^3$. Hypersurfaces have codimension one in the ambient space $\mathbb{CP}^n$, meaning they have dimension $n - 1$. An example of a hypersurface is a set of the form $\{[x_1 : \cdots : x_{n+1}] \in \mathbb{CP}^n \colon a_1x_1 + \cdots + a_{n+1}x_{n+1} = 0\}$, which is just the zero set of a linear polynomial. A hypersurface like this is called a hyperplane and is analogous to a plane in $\mathbb{R}^3$.

A reasonable question to ask of the set of projective varieties is whether it is closed under unions and intersections. Although arbitrary intersections (but only finite unions) are allowable in general, we are interested only in finite unions and intersections. In particular, we need to study finite unions and intersections of hypersurfaces, which are especially easy to understand intuitively: if $V_1 = Z(p_1)$ and $V_2 = Z(p_2)$, their union is given by

$$V_1 \cup V_2 = Z(p_1 p_2).$$

This makes sense because the product $p_1 p_2$ is zero if and only if either p_1 is zero *or* p_2 is zero (the "or" operator corresponds to taking a union). Their intersection, on the other hand, is given by

$$V_1 \cap V_2 = Z(p_1, p_2).$$

This follows from our definition of a variety, of course, but again, this makes sense because in the intersection, we need both p_1 *and* p_2 to be zero (the "and" operator corresponds to taking an intersection).

The intersection of two hypersurfaces, then, is no longer a hypersurface in general because it is defined by two polynomials. For intuition's sake, we can think of intersecting two planes in $\mathbb{R}^3$ to obtain a line, which is no longer a two-dimensional surface but rather a one-dimensional object with codimension two in $\mathbb{R}^3$. Usually, the analogy in projective space holds true: the intersection of two hypersurfaces in $\mathbb{CP}^n$ has codimension two in $\mathbb{CP}^n$, and, extending this, the intersection of s hypersurfaces in $\mathbb{CP}^n$ has codimension s in $\mathbb{CP}^n$ (we say "usually" because, as one might imagine, there are exceptions to this rule in which some kind of degenerative behavior occurs).

The union of two hypersurfaces, though, is still a hypersurface, but it is in a sense different from either of the original hypersurfaces because it is the union of two distinct "components." We may think of the union of two planes in $\mathbb{R}^3$, which certainly looks much different than a single plane in that it has two parts, each of which is a plane.

This leads us naturally into the concept of irreducibility for varieties. A variety V is defined to be reducible if it can be written as the nontrivial union $V = V_1 \cup V_2$ of two distinct varieties V_1 and V_2; if V cannot be written as such a union, it is defined to be irreducible. From our discussion of the union of hypersurfaces, it follows that a hypersurface $V = Z(p)$ is irreducible if and only if its defining polynomial p has only one irreducible factor, meaning $p = q^m$ for some irreducible q. For a reducible hypersurface, we can always write it as a finite union of "factor" irreducible hypersurfaces, where each factor is an irreducible factor of the original defining polynomial p. (add note about analogy between reducibility of polyn and hypersurface)

Our final point in this section is the following fundamental theorem of algebraic geometry: if V is a projective variety of codimension one in $\mathbb{CP}^n$, then V can be written as $V = Z(p)$ for some homogeneous $p \in \mathbb{C}[x_1, \ldots, x_{n+1}]$. Thus not only do all hypersurfaces have codimension one, but all varieties of codimension one are hypersurfaces. This may seem intuitive, but it is not as trivial as it sounds. This fact will be crucial for us in proving the existence of the polynomial we're looking for.

4 The mystery polynomial

4.1 Realizations of a triangulation

Having introduced some concepts from algebraic geometry, we now resume our discussion of triangulations and our search for this mystery polynomial. The first step will be to generalize the notion of a realization of a triangulation. It is most natural to consider such a realization as a tiling of the unit square in the real plane, as has been discussed, but it is more convenient to extend the idea as follows. The first generalization is to allow the corners of the square to move arbitrarily, with the condition that they still define a parallelogram. The second generalization, which is a bit harder to visualize, is to allow the coordinates of all the vertices to take on complex values. Note that this does not mean we map the vertices to points in the complex plane—rather, we assign each vertex two coordinates, each of which is complex. The unit square in $\mathbb{R}^2$ is thereby generalized to an arbitrary parallelogram in $\mathbb{C}^2$. The areas of the triangles are still well-defined in terms of determinants in the same way, but they are now complex in general. This is made rigorous below.

Define an acceptable realization of a triangulation $\mathcal{T}$ to be a map $g\colon V(\mathcal{T}) \to \mathbb{C}^2$ satisfying the following requirements:

1. $g(1) - g(0) = g(2) - g(3)$
2. For all vertices $v, w \in V(\mathcal{T})$, if $v \sim w$, then $g(v) = g(w)$
3. For all conditions $C_i \in \mathcal{C}$, there exists a line ℓ_{C_i} such that $g(v) \in \ell_{C_i}$ for all $v \in C_i$
4. There exists at least one triangle in $\mathcal{T}$ with nonzero area

Requirement 1 ensures that the four boundary vertices form a parallelogram in $\mathbb{C}^2$. Requirements 2 and 3 give concrete meaning to the abstract notions of the equivalence relation $\sim$ and the conditions $\mathcal{C}$, respectively: $\sim$ specifies that certain boundary vertices must have the same coordinates, and $\mathcal{C}$ specifies sets of collinear vertices. Requirement 4 simply ensures that the realization is in a sense nontrivial and that some triangles have nonzero area; this requirement is helpful because, as we have seen, zero is a problematic entity in projective space. Note that beyond these requirements, we do not put any restrictions on the coordinates of the interior vertices. We do not even require, for example, that they lie within the parallelogram defined by the boundary vertices. The interior vertices, then, are truly free to lie anywhere in $\mathbb{C}^2$, except that the appropriate sets must be collinear as specified by $\mathcal{C}$.

We define $X(\mathcal{T})$ simply as the set of all acceptable realizations of $\mathcal{T}$. $X(\mathcal{T})$ can be thought of as the space of all the ways in which to give the vertices of a triangulation coordinates in $\mathbb{C}^2$ such that the coordinates satisfy the four simple requirements above.

Let us calculate the dimension of $X(\mathcal{T})$. For now, we will only consider an honest triangulation $\mathcal{T}$ so that requirements 2 and 3 above are irrelevant. A function $g \in X(\mathcal{T})$ assigns two complex coordinates to each member of $V(\mathcal{T})$, which has $k+4$ elements when $\mathcal{T}$ has k interior vertices. These coordinates are independent except that three of the boundary vertices' coordinates determine the fourth since they must form a parallelogram. Really, then, g can freely assign two complex coordinates to $k+3$ vertices, at which point the $(k+4)$th vertex's coordinates are determined. Thus $X(\mathcal{T})$ has complex dimension $2(k+3) = 2k+6$, and in fact we can think of it as isomorphic to $\mathbb{C}^{2k+6}$.

Each map $g \in X(\mathcal{T})$ specifies the coordinates of the $k+4$ vertices of $\mathcal{T}$, which in turn specify the $2k+2$ areas of the triangles in $\mathcal{T}$. Thus there is a natural map from $X(\mathcal{T})$ to $\mathbb{C}^{2k+2}$ that, given an assignment of coordinates, gives the areas. Our next goal is to study this map.

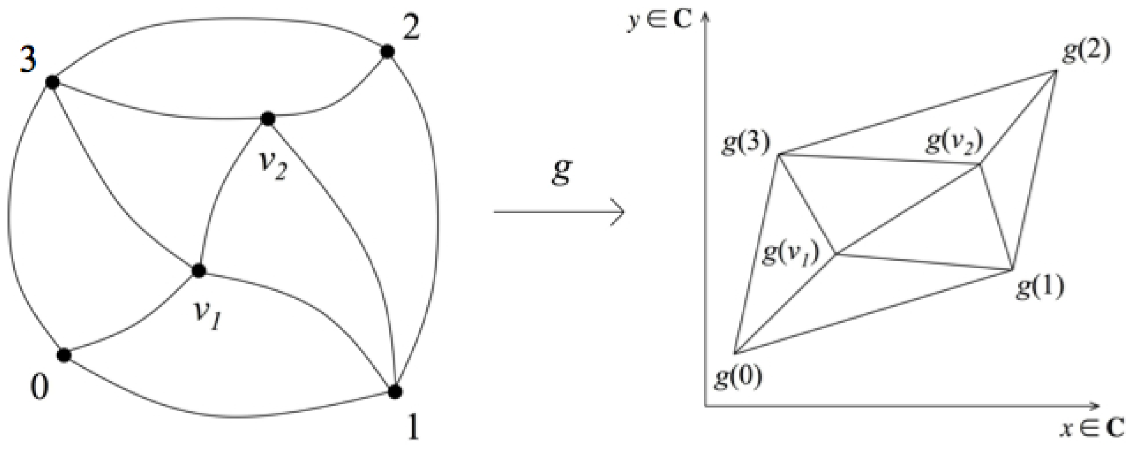


Figure 5: A generalized acceptable realization g of an abstract triangulation. Compare to Figure 4; now, the unit square can be any parallelogram, and the coordinates are complex.

4.2 The area map

As discussed above, there is a natural map from $X(\mathcal{T})$ to $\mathbb{C}^{2k+2}$ that gives the triangle areas given a particular realization of the triangulation. It will be more convenient, though, to map into projective space because the theorems of algebraic geometry are easier to apply there; this is why we have just spent so much time discussing projective varieties. Thus we compose this map giving the areas in affine space with the "projective" map from affine space into projective space. We call the resulting map f:

$$f\colon X(\mathcal{T}) \to \mathbb{CP}^{2k+1}.$$

For triangle T_i containing vertices v_a, v_b, and v_c that are mapped by $g \in X(\mathcal{T})$ to $(x_a, y_a), (x_b, y_b)$, and (x_c, y_c), respectively, Equation (2.4) gives the area of T_i. Because all the vertices, including the boundary vertices, are now allowed to vary, every area is indeed a polynomial of degree two (unless it's identically zero). The area map f specifies the *relative* areas of the $2k+2$ triangles in $\mathcal{T}$. A point in the codomain can be expressed as $[A_1 : \cdots : A_{2k+2}]$, where each A_i represents the area of a particular triangle T_i.

Let's consider the dimension of the image of $X(\mathcal{T})$ under f. f certainly can't be one-to-one since the domain $X(\mathcal{T})$ has dimension $2k+6$ and the codomain $\mathbb{CP}^{2k+1}$ has dimension $2k+1$. The "loss" in dimension is easy to explain when we recognize that f is not affected by invertible affine transformations, that is, maps formed by composing an invertible linear transformation and a translation. Formally, we may say that for any $g \in X(\mathcal{T})$, if M is an invertible 2×2 matrix and $b \in \mathbb{C}^2$, then $f(Mg + b) = f(g)$. Here $Mg + b$ is the element of $X(\mathcal{T})$ which takes the coordinates in $\mathbb{C}^2$ of each vertex as defined by g and multiplies them by M and adds b (the image of the coordinates of the triangulation under an affine transformation). Note that we must require M to be invertible ($M \in GL(2, \mathbb{C})$) to ensure that all areas aren't mapped to zero, which is not allowed in $X(\mathcal{T})$.

The set of invertible affine transformations of $\mathbb{C}^2$ (often called the affine group) has complex dimension 6: 4 from linear transformations and 2 from translations. It follows that $\dim\left(f(X\left(\mathcal{T}\right))\right) \leq 2k+6-6 = 2k$. It takes some work to prove that this is in fact an equality; for a proof, see [1]. In any case, the conclusion is that $\dim\left(f(X\left(\mathcal{T}\right))\right) = 2k$ and $f(X(\mathcal{T}))$ has codimension one in $\mathbb{CP}^{2k+1}$, a very important fact. By the theorem stated at the end of Section 3.3, this means that $f(X(\mathcal{T}))$ must be the solution set of some polynomial—this is our mystery polynomial.

Before discussing this further, we must generalize a bit. We conspicuously chose to consider only honest triangulations in the above discussion, ignoring conditions and the equivalence relation. Treating general triangulations will be the next step, but first it will be enlightening to discuss the areas of triangles, and specifically, how to make the area of a triangle zero.

4.3 Triangles with area zero

We are in the process of introducing an algorithm to compute the degree of the polynomial associated with a triangulation inductively. This algorithm works by relating a complicated triangulation to simpler ones, and the central action in this process is successively specifying that certain triangles have area zero. We now ask the question: when does a triangle have area zero?

The answer to our question is simple: a triangle has zero area if and only if its three vertices are collinear—that is, if and only if the three vertices lie on a single line. This makes sense intuitively, but we shall prove it formally.

Consider three points $(x_1, y_1), (x_2, y_2)$, and (x_3, y_3) in $\mathbb{C}^2$. Recall that the area of the triangle specified by these points is given by half of the following determinant (we can ignore the order of the vertices since we are only concerned with when the determinant is zero):

$$D = \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix}.$$

We want to show that the points are collinear if and only if $D = 0$. First we will assume they are collinear and prove that $D = 0$. There are two cases depending on whether the points are distinct.

If two of the points are the same, collinearity is trivial since two points always lie on a line. In this case, $D = 0$ automatically since two of the columns in the matrix are the same, meaning the matrix is singular and its determinant is zero.

So we may assume the three points $(x_1, y_1), (x_2, y_2)$, and (x_3, y_3) are distinct. In this case, they are by definition collinear if and only if the following condition holds: $x_3 - x_1 = c(x_2 - x_1)$ and $y_3 - y_1 = c(y_2 - y_1)$ for some $c \in \mathbb{C}$. Expand D to obtain

$$\begin{aligned} D &= x_2y_3 - x_3y_2 + x_3y_1 - x_1y_3 + x_1y_2 - x_2y_1 \\ &= x_3(y_1 - y_2) + y_3(x_2 - x_1) + x_1y_2 - x_2y_1 \end{aligned}$$

Now substitute expressions for x_3 and y_3 as given by the collinearity condition:

$$\begin{aligned} D &= [x_1 + c(x_2 - x_1)](y_1 - y_2) + [y_1 + c(y_2 - y_1)](x_2 - x_1) + x_1y_2 - x_2y_1 \\ &= 0, \end{aligned}$$

as desired.

Now suppose $D = 0$; we want the three points to be collinear. The columns of the matrix are linearly dependent, meaning we can write

$$a_1 \begin{pmatrix} 1 \\ x_1 \\ y_1 \end{pmatrix} + a_2 \begin{pmatrix} 1 \\ x_2 \\ y_2 \end{pmatrix} + a_3 \begin{pmatrix} 1 \\ x_3 \\ y_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

for some coefficients a_1, a_2, and a_3, not all zero. This is equivalent to the linear system

$$\begin{aligned} a_1 + a_2 + a_3 &= 0 \\ a_1x_1 + a_2x_2 + a_3x_3 &= 0 \\ a_1y_1 + a_2y_2 + a_3y_3 &= 0 \end{aligned}$$

If any of the points are the same, they are automatically collinear, so assume the points are distinct (in this case we may use the condition stated above). This implies that in fact none of the coefficients are zero, and we can rescale them to rewrite the above system of equations as

$$\begin{aligned} b_1 + b_2 + 1 &= 0 \\ b_1x_1 + b_2x_2 + x_3 &= 0 \\ b_1y_1 + b_2y_2 + y_3 &= 0 \end{aligned}$$

The first equation allows us to write $1 + b_1 = -b_2$; substituting this into the second and third equations and rearranging, we obtain that

$$\begin{aligned} x_3 - x_1 &= -b_2(x_2 - x_1) \text{ and} \\ y_3 - y_1 &= -b_2(y_2 - y_1), \end{aligned}$$

which is precisely the collinearity condition with $c = -b_2$. This completes the proof.

We have just seen that there are two ways to force the area of a triangle to be zero: we can specify that the three vertices are collinear, without saying anything about whether they're distinct; or we can specify that two of the vertices are equal to each other. The second condition is in a sense a special case of the first, but the two conditions are certainly conceptually distinct, and we will need to think of them as distinct in the context of the algorithm.

In defining $X(\mathcal{T})$, the set of realizations of a triangulation, it was stated briefly that the conditions $\mathcal{C}$ are "realized" as sets of points that must be collinear; that is, the coordinates of all the vertices in each condition must lie on a line in $\mathbb{C}^2$. The significance of the conditions is therefore the following: if a triangle appears in a condition, that means its three vertices are collinear for every possible realization in $X(\mathcal{T})$, so the area of the triangle as defined by f is always zero. Essentially, then, the conditions specify which triangles have zero area; modification of the conditions, which is associated with specifying that certain triangles have zero area, will play a key role in our eventual degree-computing algorithm.

Note that although each condition in $\mathcal{C}$ is really a set of triangles, there is also a set of vertices associated to it (all the vertices that are contained in one of the triangles), and the effect of the conditions on the realization of a triangulation is defined in terms of those vertices. We specify that all the vertices in each particular condition must be collinear. In fact, then, the conditions may affect more triangles than expected: if the three vertices of a triangle appear in a single condition (expressed as a set of vertices), that triangle must have area zero, regardless of whether the triangle itself appears in the condition (expressed as the original set of triangles). If a triangle is not intended to be "killed" but its area is forced to be zero in this way, we call the phenomenon "collateral damage." Figure 6 illustrates how this might happen; basically, a triangle is surrounded in some way by killed triangles, forcing its area to be zero. This is something we want to avoid in executing the algorithm.

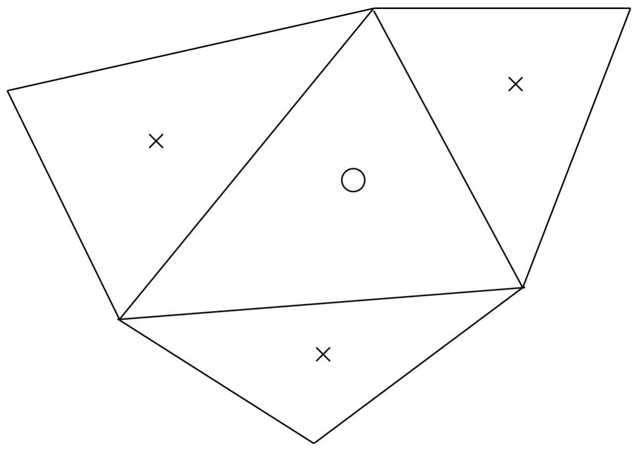


Figure 6: An illustration of collateral damage; if we kill the three outer triangles, the inner triangle must also have area zero.

The equivalence relation $\sim$ is also related to the areas of triangles. $\sim$ was defined in an abstract way, but essentially it affords a way to keep track of which boundary vertices are "equal." We specify that, for boundary vertices v and w in $V(\mathcal{T})$, if $v \sim w$, then for all realization maps g in $X(\mathcal{T})$, $g(v) = g(w)$. Because the boundary always maps to a parallelogram, $\sim$ has the ability to specify that $[g(0) = g(1)$ and $g(2) = g(3)]$ and/or $[g(1) = g(2)$ and $g(0) = g(3)]$; that is, that one or both of the opposite boundary edge pairs "collapses" into a a pair of points, meaning the boundary of the parallelogram consists of a single edge or a single point. If $0 \sim 1$ and $2 \sim 3$ (we can't have one without the other), then the edges $\{0, 1\}$ and $\{2, 3\}$ each degenerate to a single point. Thus the two triangles containing one of those two edges (there can only be exactly two) must have area zero. Similarly, if $1 \sim 2$ and $0 \sim 3$, then the triangle containing the edge $\{1, 2\}$ and the triangle containing the edge $\{0, 3\}$ must have area zero.

4.4 The area map, generalized

We now continue the discussion of the area map in the more general setting in which we consider an honest triangulation $\mathcal{T}$ along with conditions $\mathcal{C}$ and equivalence relation $\sim$. As before, we define the map f from $X(\mathcal{T})$ to $\mathbb{CP}^{2k+1}$ that specifies the relative areas of each triangle given a realization of $\mathcal{T}$. We saw that for an honest triangulation, the dimension of the image of $X(\mathcal{T})$ under f has dimension $2k$. If the conditions and equivalence relation are nontrivial, though, this is no longer the case. The situation is not much more complicated, but it requires treatment.

The effect of $\mathcal{C}$ and $\sim$ is to require that certain triangles have zero area; this clearly effects the image of $X(\mathcal{T})$. Formally, if the set of triangles that $\mathcal{C}$ and $\sim$ require to have area zero is $\{T_{i_1}, \dots, T_{i_r}\}$ and the area of triangle T_j is A_j, then $f(X(\mathcal{T}))$ is contained in the variety in $\mathbb{CP}^{2k+1}$ where $A_{i_l} = 0, 1 \leq l \leq r$. We denote this variety by $Y(\mathcal{T})$:

$$Y(\mathcal{T}) = \{[A_1 : \cdots : A_{2k+2}] \in \mathbb{CP}^{2k+1} | A_{i_l} = 0, 1 \leq l \leq r\}.$$

For a general triangulation, then, f is really a map from $X(\mathcal{T})$ to $Y(\mathcal{T})$:

$$f \colon X(\mathcal{T}) \to Y(\mathcal{T}) \subset \mathbb{CP}^{2k+1}.$$

By its definition, $Y(\mathcal{T})$ is a variety in $\mathbb{CP}^{2k+1}$ of dimension $2k+1-r$, where r triangles must have area zero; a dimension is lost for each "killed" triangle. Furthermore, $Y(\mathcal{T})$ is just a linear subspace of $\mathbb{CP}^{2k+1}$ consisting of the intersection of r hyperplanes, so in fact $Y(\mathcal{T})$ is isomorphic to $\mathbb{CP}^{2k+1-r}$.

Killing triangles (and thereby adding conditions) also must have an effect on $X(\mathcal{T})$. Whereas we saw that $X(\mathcal{T})$ has dimension $2k+6$ for an honest triangulation, this is not the case for a general triangulation. Essentially, though, we want $X(\mathcal{T})$ to be affected in the same way that $Y(\mathcal{T})$ is, meaning a single dimension is lost for each killed triangle. But the effect on $X(\mathcal{T})$ of killing a triangle is not quite as simple as that on $Y(\mathcal{T})$, which is simply a linear subspace of $\mathbb{CP}^{2k+1}$. Requirement 3 in the definition of $X(\mathcal{T})$ (Section 4.1) specifies that the vertices contained in each condition must lie on a line, which, we have seen, is equivalent to requiring that a degree-2 polynomial representing a certain determinant be zero. This, unfortunately, is not a linear condition, meaning we cannot just intersect a linear hyperplane with $X(\mathcal{T})$ each time we add a condition to kill a triangle. It is, however, a quadratic condition, meaning the situation is not much more complicated. Every time we kill a triangle, we impose a new quadratic condition on $X(\mathcal{T})$, meaning $X(\mathcal{T})$ is just the intersection of r quadratic hypersurfaces. We should expect, then, that $X(\mathcal{T})$ loses a single dimension for each of the r killed triangles.

We can also construct a more intuitive argument to imagine why we should lose a single dimension for each killed triangle. Consider a triangulation $\mathcal{T}$ containing the triangle $T_{abc} = \{v_a, v_b, v_c\}$. With no conditions, the three vertices in T_{abc} are free to move indepently in realizations of $\mathcal{T}$, and together they possess six complex degrees of freedom, since each vertex is assigned two complex coordinates. When we impose the condition that v_a, v_b, and v_c are collinear, which is equivalent to killing T_{abc}, we restrict these degrees of freedom. Essentially we can place two of the vertices anywhere we like, accounting for four degrees of freedom, but the third is no longer free. It must lie on the line joining the first two, which is essentially a one-dimensional object; we can only specify where along this line it lies. Thus the three vertices now have five degrees of freedom, one less than before, and the coordinate space $X(\mathcal{T}')$ for the triangulation $\mathcal{T}'$ created from $\mathcal{T}$ by killing T_{abc} has dimension one less than the dimension of $X(\mathcal{T})$.

It takes some significant work to prove that this extends to any number r of killed triangles, and furthermore, to prove that this loss in dimension is preserved by f, which is really what we need. The proof of this is beyond our scope here, but the result [1] is the following: the closure of $f(X(\mathcal{T}))$ is a codimension-1 subvariety of $Y(\mathcal{T})$, which is just a copy of $\mathbb{CP}^{2k+1-r}$. As a result, $\overline{f(X(\mathcal{T}))}$ can be thought of as a codimension-1 variety in $\mathbb{CP}^{2k+1-r}$, and by the theorem stated in Section 3.3, $\overline{f(X(\mathcal{T}))}$ again must be the zero set $Z(p) \subset Y(\mathcal{T})$ of some homogeneous polynomial

p. Note that p is a polynomial in $2k+2-r$ variables (the areas of the triangles that haven't been killed), but we can also think of it as a polynomial in the $2k+2$ variables $A_1, \ldots, A_{2k+2}$ representing the areas of all $2k+2$ original triangles. We should just remember that r of those variables (representing the killed triangles) never appear in any term.

This is an important moment, and it is worth recapping what we have done so far. We have shown that for every combinatorial triangulation triple $(\mathcal{T}, \mathcal{C}, \sim)$, there is a unique homogeneous polynomial equation $p = 0$ that the areas of the triangles always satisfy in any realization of the triangulation in $\mathbb{C}^2$. We will call this polynomial $p(\mathcal{T})$, or sometimes just p. Having shown the existence of $p(\mathcal{T})$, the remainder of this paper will focus on studying it.

4.5 A general discussion of $p(\mathcal{T})$

Little, in fact, is known about $p(\mathcal{T})$ for a general $\mathcal{T}$. Depending on $\mathcal{T}$, $p(\mathcal{T})$ could be quite simple or quite complex. We have already seen an example of a very simple polynomial associated to a very simple triangulation. For a more complicated triangulation containing many triangles, though, the polynomial could be much more complicated. In theory, if $\mathcal{T}$ contains n triangles and $p(\mathcal{T})$ has degree d, $p(\mathcal{T})$ could contain up to $\binom{n+d-1}{d}$ terms (the number of monomials of total degree d in n variables). Of course, there could also be fewer terms, but the problem of computing just a fraction of this number of terms seems a daunting one.

All is not lost, though—there is, in fact, a way to compute the polynomial $p(\mathcal{T})$ associated to a general $\mathcal{T}$. We actually have all the necessary information: if we assign coordinates $(x_i, y_i) \in \mathbb{C}^2$ to each vertex $w_i \in V(\mathcal{T})$, we can write explicit formulas for the area A_j of each triangle T_j in $\mathcal{T}$ in terms of the x_i's and y_i's, thereby obtaining a system of degree two polynomial equations giving each A_j. All that remains is to eliminate the x_i's and y_i's from this system to obtain a polynomial equation relating the A_j's—this equation must be equivalent to $p(\mathcal{T}) = 0$, where $p(\mathcal{T})$ is, of course, the unique polynomial we're looking for. In theory, this elimination can be done by hand, although computer algebra software can do it much more quickly. As the number of vertices and triangles increases, though, this task of elimination tends to become quite computationally expensive. Suffice it to say that for reasonably complicated triangulations, the task is realistically impossible even for our fastest computers.

We now focus, then, on a somewhat simpler task: computing only the degree of $p(\mathcal{T})$ for a general $\mathcal{T}$ and classifying triangulations according to the degree of their polynomials. We denote the degree of $p(\mathcal{T})$ by $d(\mathcal{T})$; often we may also refer to $d(\mathcal{T})$ as the degree of the triangulation $\mathcal{T}$ itself. This degree, being just a single integer, is a much simpler object of study than the whole polynomial itself. The remainder of the paper is devoted to developing an algorithm to compute $d(\mathcal{T})$ by relating it to the degrees of polynomials of simpler triangulations.

It is worth noting that Joshua Kantor and Maksim Maydanskiy [4] have completely classified all the triangulations for which $d(\mathcal{T}) = 1$; that is, they have identified necessary and sufficient conditions on $\mathcal{T}$ for $p(\mathcal{T})$ to be linear. These conditions will be discussed in Section 5.4. In the meantime, we begin to develop the algorithm.

5 The algorithm

5.1 Reducibility

We now wish to introduce the concept of reducibility for a triangulation. It is a similar notion to that of reducibility for a polynomial: we want to know whether a triangulation, with conditions, can be split into simpler component triangulations that make up the whole when taken together—"multiplied" together, in a sense. If it can be split in this way, we say it is "reducible;" whether it's reducible depends essentially on overlaps between conditions, which, remember, are related to collinear vertices and to areas. Below we define reducibility formally, and then we elaborate on the central intuition connecting reducibility and collinearity.

To define reducibility, we must unfortunately add one small piece of information to the definition of a triangulation, confusing the already too abstract definition even further. A general triangulation will now consist of an honest triangulation $\mathcal{T}$, a set of conditions $\mathcal{C}$, and an equivalence relation $\sim$ (we have already introduced these), along with two "flags"—one called the "vertical flag" and one the "horizontal flag." Each flag is just a boolean variable that can take on two values: true or false, 1 or 0, or on or off. Rather than using "true" or "false," etc., we will specify whether each flag is "raised" or not, in a rather colloquial notational convention. The two flags will be related to collinearities (conditions) involving the top and bottom edges and the left and right edges, respectively. This may seem odd and abstract, but we hope it will become clear soon enough.

Now, on to the definition: let $\mathcal{T}$ be a triangulation accompanied by conditions $\mathcal{C}$, equivalence relation $\sim$, and specified horizontal and vertical flags. $\mathcal{T}$ is defined to be reducible if there exist distinct conditions $C_1, C_2 \in \mathcal{C}$ (where the conditions are expressed as sets of vertices) for which any of the following requirements hold:

1. $|C_1 \cap C_2| \geq 2$; that is, for distinct vertices u and v, we have $u, v \in C_1$ and $u, v \in C_2$

2. $0, 1 \in C_1$ *and* $2, 3 \in C_2$ *and* the horizontal flag is not raised *and* $0 \nsim 1$

3. $1, 2 \in C_1$ *and* $0, 3 \in C_2$ *and* the vertical flag is not raised *and* $0 \nsim 3$

$\mathcal{T}$ is defined to be irreducible if it satisfies none of these.

Each of the three requirements essentially involves some kind of overlap between conditions in $\mathcal{C}$. Requirement 1 is the "primary" way in which a triangulation can be reducible; if a triangulation satisfies 1, it simply means that two of its conditions overlap in at least two vertices. Requirements 2 and 3 are an extension of this idea to the boundary vertices; we essentially want to identify the opposite edge pairs as if the triangulation lives on a torus. Triangles can then in a sense "overlap" along the boundary edges.

Reducibility, then, has to do with overlap in conditions. Let us relate this to the connection between conditions and collinearity. Suppose $C_1 = \{x, u, v\}$ and $C_2 = \{y, u, v\}$ are two conditions that both contain the vertices u and v. In a realization of these conditions, we must have that x, u, and v are collinear, and that y, u, and v are collinear. This is equivalent to requiring that x lie on the line joining u and v and that y lie on the line joining u and v, but there is only one such line! Thus both x and y lie on the line joining u and v, and in fact all four of x, y, u, and v must be collinear—that is, unless the u and v are the same point in the realization $(g(u) = g(v))$. If this is the case, then actually x and y are completely free. This is a subtle but important second case.

We identify opposite edge pairs and count them as "overlapping" as well because, again, there are two different ways in which a set of conditions like this can be achieved. For example, if $C_1 = \{x, 0, 1\}$ is a condition containing the "bottom" edge $\{0, 1\}$ and $C_2 = \{y, 2, 3\}$ is a condition containing the "top" edge $\{2, 3\}$, one way to achieve the collinearities required by C_1 and C_2 is to set $0 \sim 1$ and $2 \sim 3$, meaning that in all realizations, the bottom edge and top edge "degenerate" to a single point. Of course, the other way to achieve these collinearities is to proceed as usual and require that x lie on the bottom edge and y on the top edge. In this case we want to raise the horizontal flag to indicate that this has already been reduced and should not be counted as reducible later (we use the word "horizontal" since the top and bottom edges can be thought of horizontal, at least when the triangulation is thought of as living in a square in the plane). The situation is analogous for conditions involving the "left" edge $\{0, 3\}$ and the "right" edge $\{1, 2\}$; in that case the vertical flag is involved. The flags, then, exist simply to identify whether overlap involving boundary edges counts toward being reducible or not; a flag being raised means that the triangulation is irreducible, at least with respect to the corresponding boundary edges.

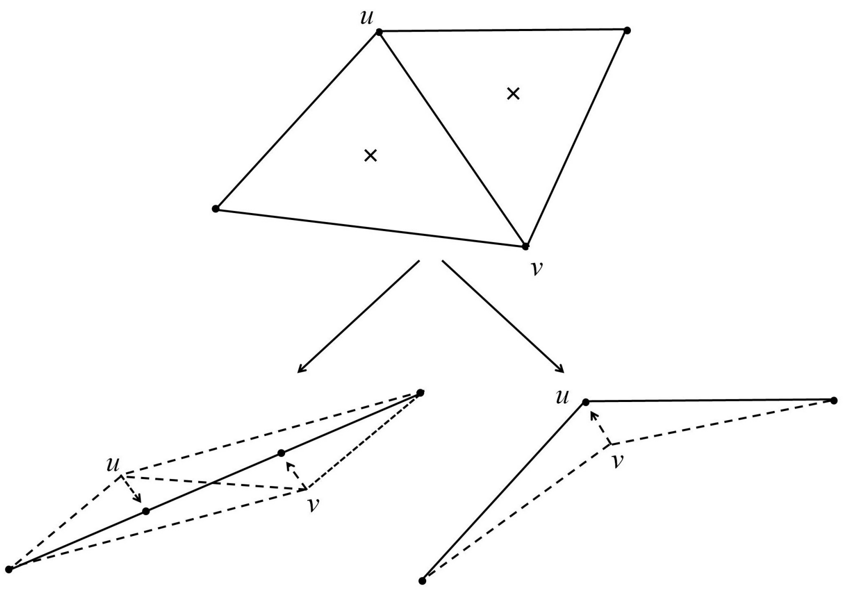
In any case, the overlap of two conditions in more than one vertex can be reduced in the sense that satisfying the conditions in a realization can happen in either of two combinatorially distinct ways. These two ways can each be represented by a different triangulation, each of which can be thought of as a component or "factor" of the original (see Figure 7). Since irreducible triangulations are more convenient, it will be useful to develop a method for factoring general reducible triangulations into a number of irreducible factors. We call this process "reducing;" the section that follows describes the process.

5.2 Reduction algorithm

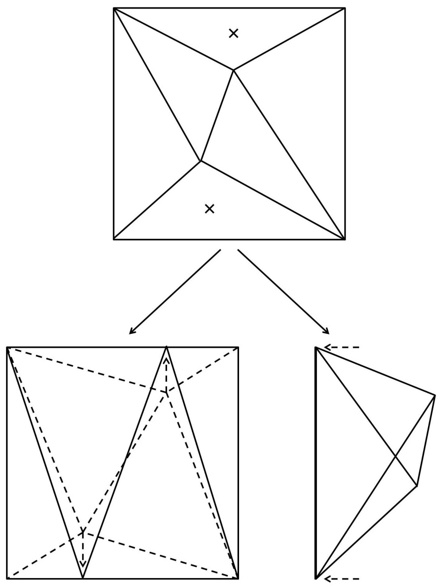
We now develop an algorithm for "factoring" a triangulation into irreducible component triangulations. Let $\mathcal{T}$ be a triangulation with conditions $\mathcal{C} = \{C_k\}_{k=1}^N$, equivalence relation $\sim$, and specified flags. Assume $\mathcal{T}$ is reducible, meaning there exist a pair of conditions C_i and C_j in $\mathcal{C}$ that satisfy one of the requirements above. If there are multiple such pairs of conditions, we can just choose one pair arbitrarily, but in practice this is not usually a problem because the algorithm works by adding a single condition (by killing a triangle) to an irreducible triangulation. Thus the ways in which a triangulation we encounter is reducible are usually quite simple. We have three cases determined by which of the three requirements C_i and C_j satisfy. In each case, we create two "factor" triangulations as described by the following:

1. If C_i and C_j satisfy requirement 1, let u and v be the vertices contained in both C_i and C_j. The factors are given by

   (a) Let $\mathcal{C}' = \mathcal{C} - C_1 - C_2 + (C_1 \cup C_2)$ (combine C_1 and C_2 into one condition). The new factor is given by the original $\mathcal{T}$, $\sim$, and flags, with conditions



(a) An edge $\{u, v\}$ is shared by two killed triangles. There are two ways to achieve this: we can make all four points collinear, or we can specify that u and v have the same coordinates.



(b) When we have the "overlap" of opposite boundary edges in the conditions, we can either proceed as normal with the flag raised or we can collapse those boundary edges into a pair of points.

Figure 7: When two killed triangles overlap on an edge, we get two distinct "factors."

$\mathcal{C}'$.

   (b) Let $\mathcal{T}'$ be $\mathcal{T}$ and let $\mathcal{C}'$ be $\mathcal{C}$ with the stipulation that all occurrences of v are replaced by u, all triangles in $\mathcal{T}$ containing both u and v are deleted, and all conditions in $\mathcal{C}$ containing fewer than three vertices are deleted, since the collinearity of two points is trivial. The new factor is given by $\mathcal{T}'$ and $\mathcal{C}'$ with the original $\sim$ and flags.

2. If C_i and C_j satisfy requirement 2, assume without loss of generality that $0, 1 \in C_i$ and $2, 3 \in C_j$. The factors are given by

   (a) Leave $\mathcal{T}$, $\mathcal{C}$, $\sim$, and the vertical flag unchanged, but "raise" the horizontal flag.

   (b) Leave $\mathcal{T}$, $\mathcal{C}$, and the flags unchanged, but set $0 \sim 1$ and $2 \sim 3$.

3. If C_i and C_j satisfy requirement 3, assume without loss of generality that $0, 3 \in C_i$ and $1, 2 \in C_j$. The factors are given by

   (a) Leave $\mathcal{T}$, $\mathcal{C}$, $\sim$, and the horizontal flag unchanged, but "raise" the vertical flag.

   (b) Leave $\mathcal{T}$, $\mathcal{C}$, and the flags unchanged, but set $0 \sim 3$ and $1 \sim 2$.

For a reducible $\mathcal{T}$, these rules give two factor triangulations. Figure 7 gives an idea of what these factors look like for each of cases 1 and 2 above. Case 3 is analogous to case 2.

In general, the two factors may not themselves be irreducible; if this is the case, we simply do it all again, factoring each of the factors into "subfactors." These subfactors are themselves factors of the root triangulation $\mathcal{T}$. Eventually, of course, we want this process to terminate. To see that it must terminate, define the integer quantity $N(\mathcal{T}) = c + v - f$, where c is the number of conditions, v the number of distinct vertices, and f the number of flags set (by definition no greater than two) in a triangulation $\mathcal{T}$. Note that every time we factor a triangulation $\mathcal{T}_0$ into two factors $\mathcal{T}_1$ and $\mathcal{T}_2$, we have that $N(\mathcal{T}_1)$ and $N(\mathcal{T}_2)$ are each strictly less than $N(\mathcal{T}_0)$. The number of vertices v is finite, so if we are able to keep factoring a triangulation, eventually the number of conditions c in each factor must go down. Eventually, then, every factor must become irreducible at the point when there are no conditions left, if not before. At this point we will have compiled a list consisting only of irreducible factors of our original reducible triangulation.

5.3 The irreducibility theorem and the foundations of the algorithm

Irreducibility of triangulations is such a nice property because of the following fact: the polynomial p associated to a triangulation $\mathcal{T}$ is irreducible if and only if $\mathcal{T}$ is combinatorially irreducible as we have just defined [1]. Furthermore, the polynomials associated to the combinatorial factors of $\mathcal{T}$ correspond to the factors of p. This can be proven, but the proof is beyond the scope of this paper.

This is very important because we are interested in the degree of the polynomial p associated to $\mathcal{T}$. If $\mathcal{T}$ can be factored into simpler component triangulations, p can be expressed as the product of the polynomials associated to each of the component triangulations, with a possible correction for multiplicity, a subtle necessity. Thus, ignoring multiplicity, $d(\mathcal{T})$, the degree of $\mathcal{T}$, is simply the sum of the degrees of the polynomials associated to each of the components.

Our algorithm for computing $d(\mathcal{T})$ relies heavily on this technique of factoring reducible triangulations, but this is only useful for studying triangulations that are, in fact, reducible. To get reducible triangulations from irreducible ones, we employ another important technique that we call “killing” triangles. The idea is to require that certain triangles (which we can usually choose in an arbitrary order) must have area zero in every realization of the triangulation, a requirement that is incorporated into the conditions. After killing a triangle, we obtain a triangulation whose conditions stipulate that the triangle has area zero; this new triangulation can be thought of as a special case of the original, and its polynomial is closely related to the polynomial of the original. By successively killing triangles, we change the polynomial, but we change it in a predictable way. Because we are changing the conditions, we also eventually end up with a reducible triangulation, which can be reduced into simpler factors, whose polynomials' degrees basically sum to the degree of the original polynomial.

The theoretical basis that allows killing triangles to be useful is contained in the following: suppose $\mathcal{T}$ is a triangulation with conditions $\mathcal{C}$. Let $T_i = (u, v, w)$ be a triangle in $\mathcal{T}$. We want to kill T_i; this means we specify that T_i has area zero, which happens if and only if u, v, and w are collinear. Thus killing T_i is equivalent to adding the condition $C = \{u, v, w\}$, meaning, really, that u, v, and w are collinear, to $\mathcal{C}$. Let $\mathcal{C}' = \mathcal{C} \cup C$, and let $\mathcal{T}'$ denote the triangulation $\mathcal{T}$ with conditions $\mathcal{C}'$. There is some polynomial $p = p(\mathcal{T})$ associated to $\mathcal{T}$ and another polynomial $p' = p(\mathcal{T}')$ associated to $\mathcal{T}'$. We are interested in p, and we want to relate it to p'.

p is a homogeneous polynomial in $A_1, \ldots, A_{2k+2}$, where $\mathcal{T}$ has k interior vertices. From p, we can construct another polynomial by discarding all terms in which A_i (representing the area of T_i) appears. In doing so, we are effectively specifying that $A_i = 0$. Call the resulting polynomial $p|_{A_i=0}$. Significantly, since p is homogeneous, meaning every term in p has the same degree, it follows that $p|_{A_i=0}$ and p have the same degree.

Now, we must notice that $\mathcal{T}'$ is really just a special case of $\mathcal{T}$ in which A_i is always zero. This means that if in some realization of $\mathcal{T}'$, we have that the areas $A_1, \ldots, A_i = 0, \ldots, A_{2k+2}$ satisfy $p' = 0$, then they also satisfy $p|_{A_i=0} = 0$. Our intuition might tell us that this means p' divides $p|_{A_i=0}$, but this is not quite correct. In fact p' may not be irreducible; if not, though, we can factor it as $p' = q_1 \ldots q_r$, where each q_i is an irreducible polynomial. Again we have that $q_i = 0$ implies $p|_{A_i=0} = 0$, so since the q_i's are irreducible, we can conclude that every q_i divides $p|_{A_i=0}$. In fact, the q_i's are the only factors of $p|_{A_i=0}$, and so

$$p|_{A_i=0} = q_1^{m_1} \dots q_r^{m_r}$$

for nonnegative integers m_i's, which are the multiplicity of the respective q_i's. Let us assume for now that each multiplicity is one (which is usually the case), because in this case a quite useful conclusion can be drawn. It is that in fact $p' = p|_{A_i=0}$, and furthermore, that the degree of p' must be equal to the degree of p. By killing a triangle to obtain a new triangulation, we have not changed the degree of the associated polynomial!

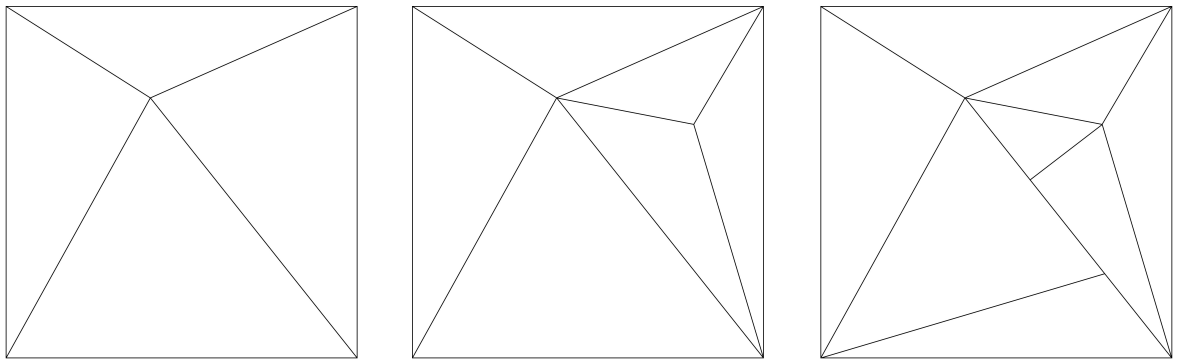
5.4 Linear triangulations

As promised, we now discuss conditions on $\mathcal{T}$ that give a linear polynomial; that is, a polynomial of the form $p(\mathcal{T}) = \sum_{i=1}^{2k+2} a_i A_i$, where $a_i \in \mathbb{C}$. Kantor and Maydanskiy [4] have proven that $p(\mathcal{T})$ is linear if and only if $\mathcal{T}$ is of one of the following forms or a subtriangulation of one of the following forms:

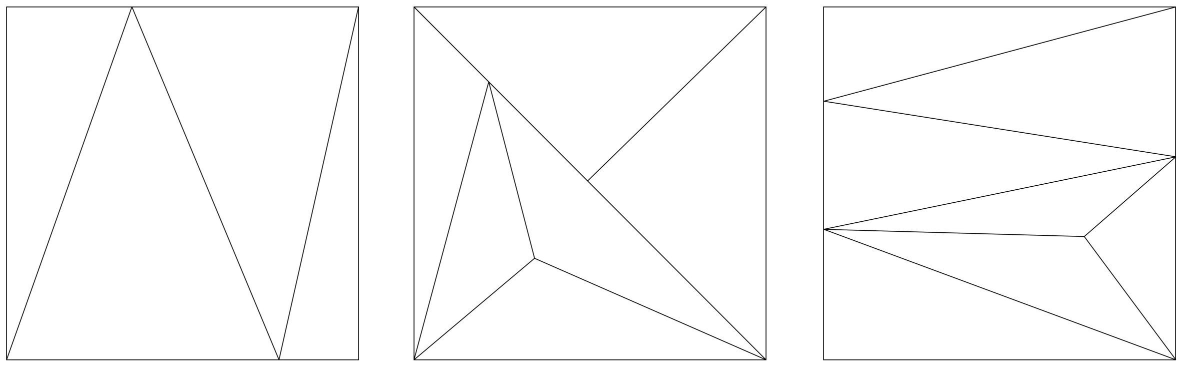
1. *"'X' case:"* There exists a vertex w such that w is connected by an edge to each of the four boundary vertices. In this case the triangulation resembles the letter 'X' when drawn in the plane.

2. *"Zigzag case:"* A "zigzag" pattern of edges is formed between two distinct boundary vertices, where the intermediate vertices are each required by a condition to lie on a boundary edge. A special case of this is the "diagonal case," in which a single diagonal edge joins two opposite boundary vertices (0 and 2 or 1 and 3).

These are best illustrated by a diagram; Figure 8 provides examples of each case. Note that a subtriangulation of a triangulation of $\mathcal{T}$ is defined as any triangulation $\mathcal{T}'$ that can be obtained from $\mathcal{T}$ essentially by adding vertices and edges; $\mathcal{T}'$ basically contains $\mathcal{T}$ plus some extra information. This effectively means that the "edges" in each of the two cases can consist of either actual edges of triangles or of sets of mutually collinear vertices (conditions). Figure 8 also provides example of subtriangulations of the two linear cases.



(a) 'X' case.



(b) Zigzag case.

Figure 8: Examples of linear triangulations, including examples of subtriangulations of the basic cases.

Given an irreducible triangulation, it is easy to check whether or not the associated polynomial is linear. Our goal in the algorithm will be to produce linear factors of an original triangulation; by counting these factors, we can count the degree of the polynomial associated to that triangulation.

The following is worth noting, if only briefly: there is a particular type of linear triangulation whose polynomial is especially interesting. Define H as the subset of $\mathbb{CP}^{2k+1}$ in which $\sum_{i=1}^{2k+2} A_i = 0$ (a hyperplane). We say that a triangulation $\mathcal{T}$ is in H if $f(X(\mathcal{T})) \subset H$; this means that all the areas of the triangles in $\mathcal{T}$ always sum to zero, and in fact $p \sum_{i=1}^{2k+2} A_i$ is the polynomial associated to $\mathcal{T}$. This p is certainly linear, and indeed all triangulations in H are special cases of either the 'X' case of the zigzag case. In our algorithm, the only way we will ever encounter triangulations in H is by setting $0 \sim 1$ and $2 \sim 3$ or $0 \sim 3$ and $1 \sim 2$, meaning that the boundary parallelogram collapses into a single line segment. The area of the parallelogram is then required to be zero, so the areas of the triangles must sum to zero. Triangulations in H are curious, but for our purposes, we will just count them as linear and be done with them.

5.5 The algorithm

All the framework is in place; now we explicitly describe the execution of the algorithm to compute $d(\mathcal{T})$.

Let $\mathcal{T}$ be a triangulation with conditions $\mathcal{C}$, equivalence relation $\sim$, and flags. If $\mathcal{T}$ is reducible, the first step is to find all the irreducible factors of $\mathcal{T}$ according to the reduction algorithm. Then we test each of these factors to determine if their polynomials are linear; we set aside the linear factors and keep track of how many of them there are with a variable d, a "degree counter." Every time a linear factor is found, d is incremented. All the remaining nonlinear factors are put into a list L. At this point, we have that $d(\mathcal{T})$ is equal to d plus the sum of the degrees of the irreducible factors in L; this relationship should hold at every step in the algorithm. The goal is to make the factors in L successively simpler by killing triangles, all the while factoring out linear factors and incrementing d, until L is empty and all the factors are linear.

The basic step of the algorithm is to choose an irreducible factor triangulation $\mathcal{T}_0$ from the list L and kill a triangle T in $\mathcal{T}_0$ to obtain a new triangulation $\mathcal{T}_0'$. Note that $d(\mathcal{T}_0) = d(\mathcal{T}_0')$. If $\mathcal{T}_0'$ is still irreducible, repeat this step, killing more triangles one at a time, until a reducible triangulation is obtained, whose degree will still be equal to the degree of $\mathcal{T}_0$. Then reduce this triangulation into irreducible factors, the sum of whose degrees will add to the degree of $\mathcal{T}_0$. Test each factor for linearity, and for every linear factor, increment d. Then put the remaining nonlinear factors into L to be processed later. Now L does not consist of triangulations that are factors of the original triangulation $\mathcal{T}$, but it does consist of "factors" in a loose sense, and the sum of those factors plus d is still always equal to $d(\mathcal{T})$.

This step is repeated with a new factor triangulation $\mathcal{T}_0$ taken from the updated L, and the process continues to loop. We would like this process to terminate, and in fact we can prove that it does. A sketch of the proof is the following: note that every time we kill a triangle in a factor $\mathcal{T}$ to obtain $\mathcal{T}'$, we have that $\dim(Y(\mathcal{T}')) < \dim(Y(\mathcal{T}))$. If we always choose the factor $\mathcal{T}_0$ in L with $\dim(Y(\mathcal{T}_0))$ greatest, the maximum dimension of $\dim(Y(\mathcal{T}))$ over all triangulations $\mathcal{T}$ in L will go down at every step (or after some finite number of steps). Thus after a finite number of steps, the algorithm will terminate: L will be empty, and we'll have $d = d(\mathcal{T})$—we're done!

5.6 Issues with the algorithm

We have described an algorithm for computing $d(\mathcal{T})$ for a general $\mathcal{T}$; this algorithm works by counting the degrees of a number of factors of $\mathcal{T}$. The algorithm does its job reasonably well in the sense that all the factors it counts are related to the original triangulation, and there are no factors that it "misses." Where it could go wrong, however, is in counting the multiplicity of a factor, and this could affect whether the resulting total degree counted is correct.

By the multiplicity of a factor, we essentially mean the "power" of a factor triangulation's polynomial that appears in the parent triangulation's polynomial (the m_i's in the equation at the end of Section 5.3). This is the same as the number of times that a factor triangulation should be counted toward the degree of a parent reducible triangulation. The algorithm as we have described so far operates under the assumption that all the multiplicities are one, meaning, really, that every factor we encounter in the algorithm should be counted exactly once toward the total degree. This is why we increment the degree counter d by exactly one each time a linear factor is found. In theory, though, this may be an incorrect assumption: some multiplicities may be zero, in which case certain factors should not be counted (we shouldn't increment the counter at all); or some multiplicities may be greater than one, meaning certain factors must be counted multiple times (we should increment the counter by some integer $m \geq 2$).

One way the first case can arise is when a factor triangulation contains "collateral damage," a concept that was briefly introduced earlier. We say that collateral damage occurs when a triangle is forced to have zero area in all realizations, but that triangle has not specifically been chosen to be "killed." This affects the dimension of $X(\mathcal{T})$ and the relationship between $X(\mathcal{T})$ and $Y(\mathcal{T})$. If this happens, the resulting factor should not be counted because in a sense it has zero multiplicity. Luckily, collateral damage can easily be detected: for each triangle (u, v, w) that we haven't killed yet, we simply check whether any condition contains all of u, v, and w. If so, (u, v, w) must have area zero and we discard the factor because collateral damage has occurred.

It is also possible that a factor triangulation has multiplicity higher than one, meaning its degree should be counted two or three times, for example. This is a more subtle issue than collateral damage because we know of no way to test for higher multiplicity. We believe, however, that higher multiplicities can be avoided by choosing the order in which to kill triangles in a smart way. Conspicuously, we have chosen not to discuss this order at all; this is because, if we ignore higher multiplicities, the algorithm we have described works for any arbitrary (even random) choice of order in which to kill triangles. Unfortunately, we can't always ignore higher multiplicities, so we must choose this order in a particular way. The ideal order involves a shelling of a triangulation; essentially we always want to choose a triangle that shares an edge with a triangle that has already been killed. If we do this, the triangulation will be reducible at every step, and it will always be reducible in a very simple way. We think we can prove that this method results in factors that never have multiplicity greater than one, but because this method is still in development, we will not present it in detail here. For a more detailed discussion, see [1].

The conclusion is this: with a smart choice of order in which to kill triangles and a check for collateral damage, this algorithm computes the correct degree $d(\mathcal{T})$ of the polynomial $p(\mathcal{T})$ associated to a triangulation $\mathcal{T}$.

The preceding development of the algorithm has been an end in itself, and it is nice to know that we have a method for computing the degree. This method, though, is somewhat tedious to implement by hand; for this reason, we have developed a computer program to implement it. The following section gives a summary of the steps necessary in programming the algorithm.

5.7 Implementation of the algorithm in Java

We have developed a program in Java that implements the algorithm. Rather than including all the code or even complete pseudocode, we simply describe it generally. It is important to note that although the algorithm is based upon the geometry of triangulations, all the steps of the algorithm itself are completely combinatorial and can be implemented without knowing anything about the geometry. Indeed, the program described below involves only simple combinatorial operations with sets of vertices.

The most important objects in the algorithm are triangulations and conditions, both of which involve sets of vertices. The basic operations of the algorithm, such as checking if a triangulation is reducible, reducing it into irreducible factors, and checking if a factor is linear, themselves involve a number of basic operations on these sets of vertices. These basic operations include adding and removing vertices from a set, taking the union and intersection of two sets, and checking if a vertex is contained in a set. The Java class `HashSet` provides a "set" environment in which these operations are simple to perform. To keep track of collections of sets (i.e., triangulations and conditions) as well as collections of triangulations, the class `ArrayList` is used, which provides an ordered list environment.

Vertices are represented by `Integer` objects, with the boundary vertices represented by 0 through 3 (as one might expect) and the k interior vertices by 4 through $k + 3$. An honest triangulation is represented by a list of sets of three vertices (each set representing a single triangle); specifically, a triangulation is an `ArrayList<HashSet<Integer>>` object. A set of conditions is also represented by an `ArrayList<HashSet<Integer>>` object where each `HashSet` contains no fewer than three `Integer` objects (unlike a triangulation, it can be more than three); each `HashSet`, of course, is just a single condition representing a set of vertices that must be collinear.

It is convenient to represent a general triangulation with conditions as a single object so that they may be easily manipulated together. The class `Configuration` was written for this purpose; a `Configuration` object contains two fields representing the honest triangulation and conditions, each containing an `ArrayList<HashSet<Integer>>` object. An `int` field `degreeCounter` keeps track of the number of linear factors; when the algorithm finishes, this number is the degree. There are also two `boolean` fields representing the horizontal and vertical flags. The equivalence relation $\sim$ need not be represented because whenever it becomes nontrivial, a triangulation immediately becomes linear and can be discarded.

The class `Algorithm` essentially contains the machinery for running the algorithm. A field named `toDoList` containing an `ArrayList<Configuration>` object corresponds to the list L of irreducible, nonlinear factors. A `main()` method in another application class actually runs the algorithm by manipulating an `Algorithm` object.

The `Algorithm` class defines several methods that each perform a specific operation on a `Configuration` object:

1. `isReducible()`: Returns a `boolean` value indicating whether a `Configuration` is reducible or not. It works by looping through the rows in `conditions` and taking pairwise intersections; if any intersection contains more than two elements, requirement (1) for reducibility is satisfied. The method performs a separate check for requirements (2) and (3), checking if a condition contains both 0 and 1 (0 and 3) and another contains 2 and 3 (1 and 2). If at any point a requirement for reducibility is satisfied, the loop stops and `true` is returned, indicating `Configuration` is reducible.

2. `reduce()`: Returns an `ArrayList<Configuration>` object that lists the irreducible factors of an inputted `Configuration`. During the reduction process, an intermediate `ArrayList<Configuration>` object keeps track of the reducible factors as they arise. Each reducible factor is split into two factors as described in Section 5.2, which are each either added to the list to return (if they are irreducible) or added back to the intermediate list (if they are still reducible).

3. `isLinear()`: Returns a `boolean` value indicating whether the polynomial associated to a `Configuration` is linear or not. Since there are two ways to be linear, it performs two separate checks:

    (a) `isX()`: Checks if any vertex is "connected" to all four boundary vertices by either an edge of a triangle or by a condition.

(b) `isZigzag()`: Compiles a list of vertices that lie on opposite edge pairs and searches for a “zigzag” path between these lists that starts and ends at two different boundary vertices.

If either of these returns `true`, `isLinear()` returns `true`; otherwise, it returns `false`.

4. `killTriangle()`: Kills a triangle by adding a single extra condition to a `Configuration` consisting of the three vertices of that triangle. The triangle must first be chosen in some way; this can happen through user input, randomly, or by some deterministic algorithm.

Figure 9 provides an illustration of the procedure of the algorithm in a flow chart.

5.8 Applying the algorithm to a simple example triangulation

To illustrate the algorithm at work, consider the honest triangulation represented by Figure 2. This triangulation contains two interior vertices and six triangles and is irreducible because there are no conditions. We will run through the algorithm starting with this triangulation. Figure 10 traces the triangulations obtained at each step of the algorithm, represented for convenience as triangulations of a square in the plane. An ‘x’ inside a triangle indicates that that triangle wlll be killed in the next step.

Our first step is to choose a triangle to kill. Somewhat arbitrarily, we choose $(3, 4, 5)$. After killing $(3, 4, 5)$, we of course have the single condition $\{3, 4, 5\}$, which is still irreducible.

Next we choose to kill $(4, 1, 5)$. Now we have two conditions: $\{3, 4, 5\}$ and $\{4, 1, 5\}$,

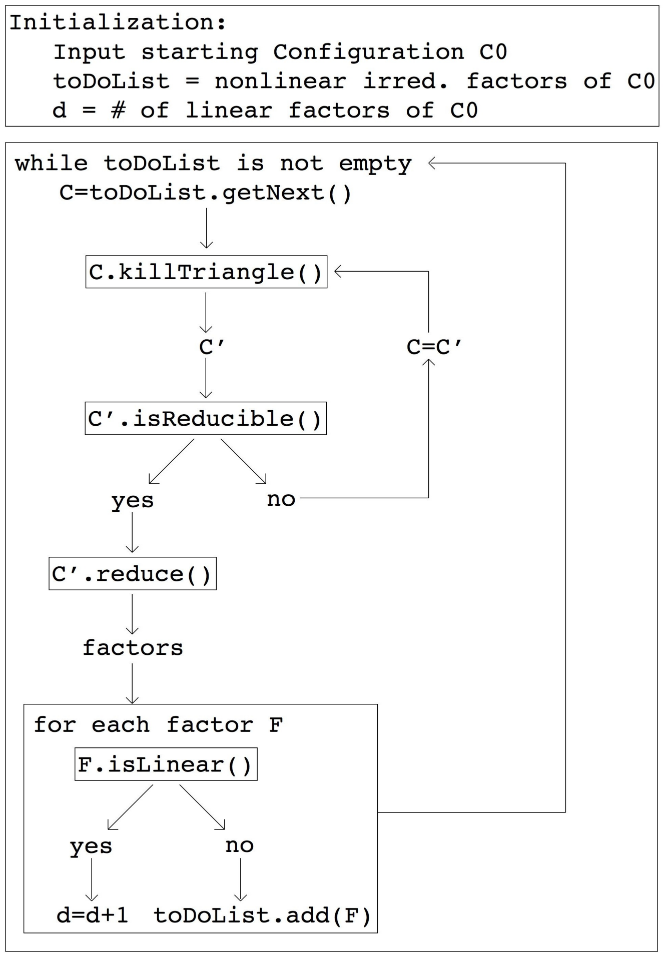


Figure 9: Flow chart illustrating the procedure of the algorithm.

which is reducible because both 4 and 5 appear in two conditions. Reducing this, we obtain two factors:

1. Set $4 = 5$ to obtain a triangulation with a single interior point and four triangles. This is a linear triangulation because it is an ‘X’ case.

2. Combine the conditions into the single condition $\{1, 3, 4, 5\}$, meaning 4 and 5 lie on the diagonal of the “square” joining 1 and 3. This is also linear because it is a subdivision of the zigzag case.

Both of the factors obtained are linear, so we have no factors left. We conclude that the original triangulation had degree two.

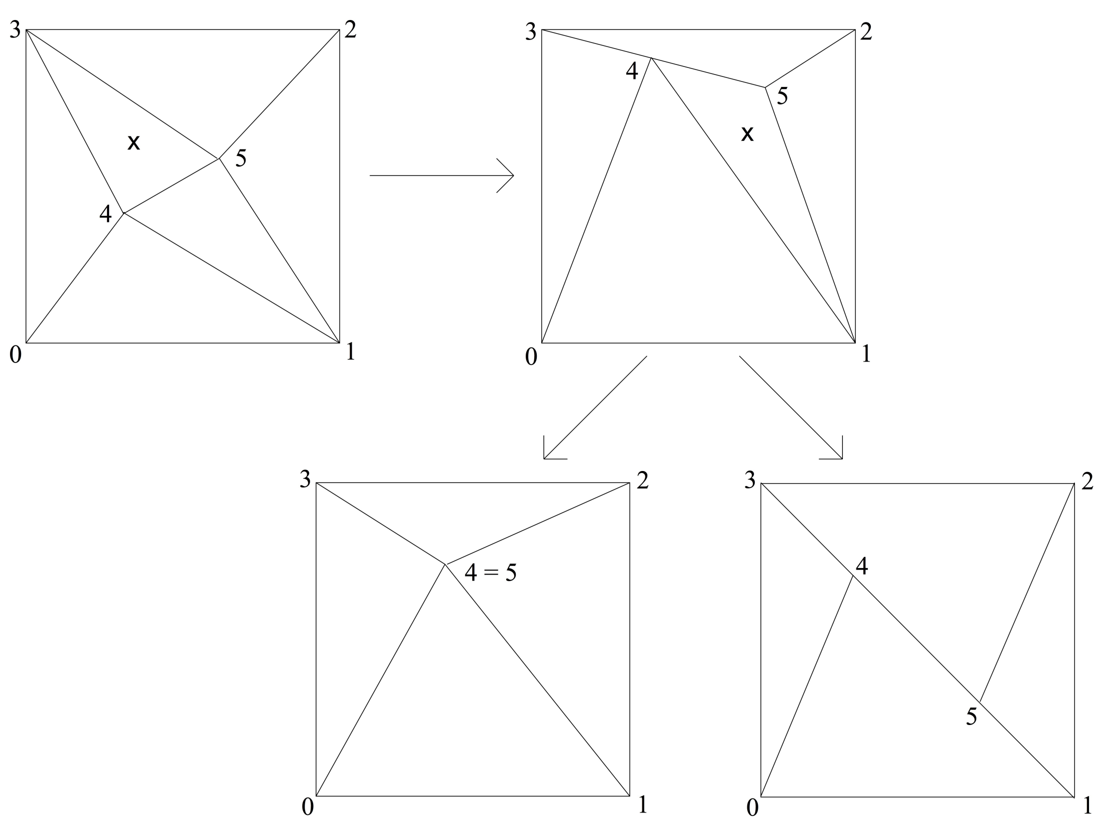


Figure 10: The stages of the algorithm for a simple triangulation.

6 Conclusion

We have completed our discussion of the algorithm to compute the degree. Let us summarize what we have done. First, we consider a triangulation as an abstract combinatorial object with a number of vertices, edges, and triangles. We want to imagine this object as living inside a square in the plane, but we must generalize a bit and actually place it inside a parallelogram in $\mathbb{C}^2$, assigning two complex coordinates to each vertex. For each assignment of coordinates, we can define the complex areas of the triangles; we then think of these areas as living in complex projective space. Using some facts from algebraic geometry, we prove that the areas of the triangles must satisfy a certain homogeneous polynomial equation $p = 0$ that depends on the combinatorics of the triangulation.

We then focus our attention on the degree of the polynomial p. We develop an algorithm to compute this degree by relating it to the degrees of simpler triangulations. These simpler triangulations are obtained from the original by “killing” triangles, or specifying that certain triangles have area zero. Killing triangles has predictable implications on the geometry of a triangulation and on the associated polynomial. At some points, multiple “factor” triangulations are obtained whose degrees sum to the degree of the root triangulation. By successively applying the algorithm to these factors, eventually we obtain a number of triangulations with linear polynomials, which are easy to identify. In the end, we essentially just count the number of such linear factors to obtain the degree of the original triangulation.

We believe that this algorithm actually gives the correct degree for any general triangulation, although the complete proof has not been presented here. Much of the work for this thesis has been in developing this algorithm, the details of which have evolved with time. The rest of the work has been in programming the algorithm in Java, a process which we have briefly described in this paper.

Beyond the proof that this algorithm always works, which is an end in itself, we also have several other general goals in mind. One is simply to classify triangulations according to the degree of their polynomials. All linear triangulations have been identified, but little is known about triangulations of higher degree. Our hope is that calculations done via the algorithm program may lead to conjectures and eventually proofs about what kind of triangulations have, for example, quadratic polynomials. There would appear to be a wealth of potential theorems to this general effect out there, waiting to be discovered.

Unfortunately, time has limited the number of such calculations we have been able to perform, and we have no such conjectures as of yet. In order to perform a number of calculations efficiently, though, the computer implementation of the algorithm and the user interface would probably have to be streamlined. Improving the program to incorporate features such as automated selection of triangles based on a shelling and an easier method for inputting starting triangulations is an immediate goal of this project.

References

[1] A. Abrams and J. Pommersheim. A polynomial invariant of square triangulations. Preprint.

[2] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra.* Springer, second edition, 1997.

[3] J. A. Gallian. *Contemporary Abstract Algebra.* Houghton Mifflin, sixth edition, 2006.

[4] J. Kantor and M. Maydanskiy. Triangles gone wild. In S. Katok, A. Sossinsky, and S. S. Tabachnikov, editors, *Mass Selecta: Teaching and Learning Advanced Undergraduate Mathematics*, pages 277–288. American Mathematical Society, 2003.

[5] K. E. Smith, L. Kahanpää, P. Kekäläinen, and W. Traves. *An Invitation to Algebraic Geometry.* Springer, 2000.