The One Hundred and Twelve Point Three Degree Theorem

Jacob Garber, Boyan Marinov, Kenneth Moore, George Tokarsky
University of Alberta

May 29, 2020

Abstract

It has been known since Fagnano in 1775 that an acute triangle always has a periodic billiard path, given by its orthic triangle. The case of obtuse triangles has proven much more difficult to solve, and so far has only been shown in a number of special cases. We give a improvement on the method of Schwartz [3], and a rigorous computer-assisted proof that a triangle has a periodic billiard path when all its angles are at most 112.3 degrees.

1 Introduction

Given a triangle, we define a *trajectory* or *pool-shot* (Figure 1.1) simply as a ray which is bent according to the following rules:

1. The ray begins on one edge of the triangle (not at a vertex).
2. The ray's direction initially points into the triangle.
3. When the ray collides with an edge, it is reflected according to the law of reflection.
4. If the ray collides with a vertex, it ends there.

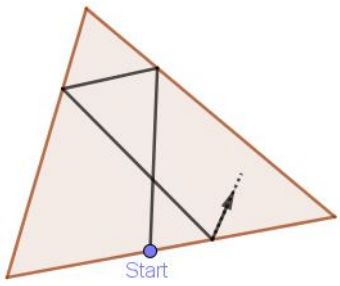


Figure 1: A Trajectory

A periodic path then is a trajectory which eventually collides with its starting point at the same angle from which it left, thus it will infinitely repeat the same finite sequence of reflections. This concept leads us to a natural idea:

Conjecture 1.1. Every non-degenerate triangle has a periodic path.

In 1993, Holt [1] showed that every right triangle has a periodic path using simple arguments. It is currently unknown whether every obtuse triangle has a periodic path, and there don't appear to be any simple arguments. In between, there have been various partial results. In 1986, Masur [2] proved that every rational obtuse triangle has a periodic path, and in 2006, Schwartz [3] showed that every obtuse triangle with obtuse angle at most 100 degrees has a periodic path using a computer assisted proof.

The aim of this paper is to show that every obtuse triangle with obtuse angle at most 112.3 degrees has a periodic path using a similar computer assisted proof.

From the work of Hooper and Schwartz [4] certain obtuse triangles contain only unstable periodic paths. These are Veech triangles of the form $(90/2n, 90/2n)$ where $n \geq 2$. A complete proof of the 180 degree theorem will thus require using unstable paths, and therefore there is no reason to exclude the use of unstable paths in our paper, and doing so would also add to the extra number of paths required. We did however extend the 100 degree Schwartz proof to 105 degrees using the 134 stable codes listed in Appendix B, and used a massive amount more paths not listed to push the result from 105 to 112.3 degrees.

112.5 degrees appears to be a natural barrier in this method, and to go further would need a new infinite family of paths. From extensively searching the paths around this area, it does not appear that there is a small collection of infinite families that will cover a semicircle around the point (0, 67.5). To cover this point may require generalizing the notion of which paths are considered to make up a family.

This paper is organized as follows. In section 2, we give an overview of notation and how we describe a path. Section 3 is about the "space of triangles", and how to strategize our brute force attack. In section 4, we give some background on "unfoldings", which is an essential concept for our proof. In section 5 we go through the classification and properties of different paths, as well as how one can prove rigorously through calculation whether a given path is periodic for a triangle. In section 6, we go through some of the previous results on special cases of this problem using our notation. In section 7 we go through the proofs of two infinite patterns of paths that are required in this scheme. Finally, in section 8 we discuss further optimizations, how the program works, and how we have made the calculations rigorous.

2 Side, Code, and Binary Sequences

2.1 Side Sequences

To begin classifying billiard trajectories, we first need some notation. Given a triangle with angles x, y, and z, we label the side opposite x with 1, the side opposite y with 2, and the side opposite z with 3. For any periodic billiard trajectory in the triangle, we then define its *side sequence* to be the list of consecutive sides that are hit during one cycle. For example, for the billiard trajectory shown in Figure 2, if we begin at side 2 and continue to side 1, it has the side sequence 213231323. We formalize this as follows.

Definition 2.1. A *side sequence* is a finite string of integers from the set $\{1, 2, 3\}$, where all adjacent integers are distinct, including the first and last. Any sequence of this form represents a potential periodic billiard trajectory.

There are other equivalent side sequences for the trajectory in Figure 2 that we could consider. Given the periodic nature of the trajectory, the initial starting side and direction are both arbitrary, so we could instead start on side 3 and continue to side 1, giving the side sequence 313231232. These choices correspond to rotations and reversals of the original side sequence, respectively. Likewise, we could continue around the triangle for as many cycles as we wish, leading to the infinite family of side sequences 313231232313231232, 31323123231323123231323123231232, etc. All of these sequences represent the same trajectory, so it is convenient to pick one among them as being canonical.

Definition 2.2. A side sequence is said to be in *standard form* if it has no proper repeating substrings and is lexicographically least among all its rotations and reversals.

All side sequences can be converted to an equivalent one in standard form. For example, the standard form of the side sequence 213231323 is 123231323.

In this way, we can uniquely identify every periodic billiard trajectory by a side sequence in standard form. The converse, however, is false. Given a side sequence, there may not exist a triangle with a periodic billiard trajectory of that type. We call such side sequences *empty*.

Lemma 2.3. All side sequences that contain only two types of sides are empty.

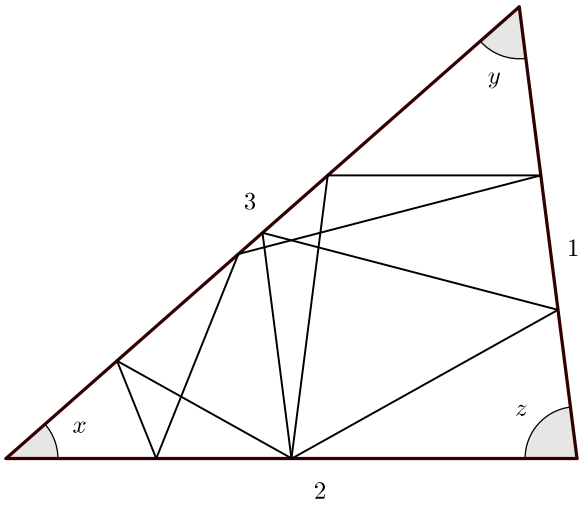


Figure 2: Periodic Billiard Trajectory 213231323

Proof. Since no two adjacent integers can be identical, including the first and last, all such sequences must be of the form $(ab)^n$ for some $n \geq 1$ and distinct $a, b \in \{1, 2, 3\}$. Removing duplicate cycles, this sequence is equivalent to ab. However, for such a sequence to be periodic and obey the law of reflection, sides a and b must be parallel, which cannot happen in a triangle. □

2.2 Code Sequences

Long side sequences quickly become cumbersome to work with, so it is helpful to introduce a more compact notation. For each pair of integers in the side sequence (including the first and last), write out the angle subtended by the billiard trajectory as it moves between those sides. For the side sequence 123231323, this is $yzxxxyyxx$. Then, group the consecutive angles together, shuffling identical angles from the back to the front if necessary, and count the number of angles in each group. For our example, this gives $1y\ 1z\ 3x\ 2y\ 2x$. The sequence of integers 1 1 3 2 2 is called the *code sequence*, and the sequence of angles $y\ z\ x\ y\ x$ the *angle sequence*. Given a code and angle sequence, it is possible to recover the original side sequence.

Note that specifying any two consecutive angles will completely determine the rest of the angle sequence. For example, for the sequence 1 1 3 2 2, consider the angles $z\ x$ for the 1 and the 3. After bouncing once across angle z, the pool ball will bounce 3 times across angle x, and 3 being odd, will end at the opposite side from where it started. It will then bounce twice across angle y, and 2 being even, will end up at the same side as where it started, and then bounce back across the previous angle x. Continuing this pattern will produce the rest of the angles in the angle sequence and, crucially, wrap-around to match the $z\ x$ we started with. We can use this wraparound property to check if a list of integers represents a valid code sequence, which we formalize in Algorithm 2.1.

Definition 2.4. A *code sequence* is a non-empty list of positive integers that is valid by Algorithm 2.1. As with side sequences, a code sequence is in *standard form* if it has no proper repeating sub-code sequences, and is lexicographically least among its rotations and reversals. For example, the standard form of 1 1 3 2 2 is 1 1 2 2 3.

Using this property, we define the *initial angles* to be the first two angles of a code sequence. (Code sequences of length 1 we can ignore, since these only contain two sides and are empty by Lemma 2.1.) There are six possible angle pairs: $x\ y$, $x\ z$, $y\ x$, $y\ z$, $z\ x$, $z\ y$. These correspond to the six possible relabelings of the angles of a triangle. (Mention relabelings with side sequences?)

Algorithm 2.1 Valid Code Sequences

```
1: function OtherAngle(prev, curr)
2:     match (prev, curr)
3:         case (x, y) — (y, x): return z
4:         case (x, z) — (z, x): return y
5:         case (y, z) — (z, y): return x
6:     end match
7: end function

8: function Valid(sequence)
9:     prev, curr := x, y              ▷ The angles here are arbitrary, so long as they wrap around correctly
10:    for num in sequence
11:        next := if num % 2 = 0 then prev else OtherAngle(prev, curr)
12:        prev, curr := curr, next
13:    end for
14:    return prev = x and curr = y
15: end function
```

2.3 Binary Sequences

In Algorithm 2.1, the validity of a code sequence only depends on the parity of its numbers. For a code sequence, we thus define its corresponding *binary sequence* to be each code number modulo 2. For instance, the binary sequence of 1 1 2 2 3 is 11001. We can treat binary sequences as a formal language, and using the six angle pairs mentioned above, convert Algorithm 2.1 into the following six-state finite automaton.

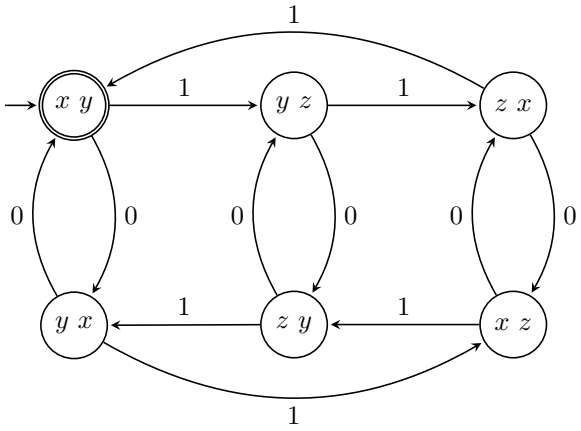


Figure 3: Finite Automaton for Binary Sequences

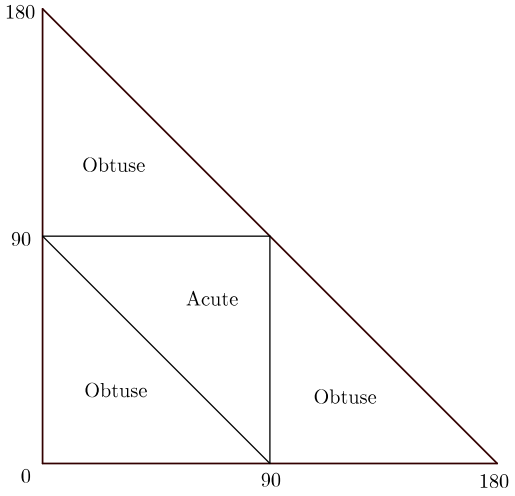
Definition 2.5. A *binary sequence* is a finite binary list that is accepted by the automaton in Figure 3. A binary sequence is said to be in *standard form* if it has no proper repeating sub-binary sequences, and is lexicographically least among all its rotations and reversals. For example, the standard form of 11001 is 00111.

Corollary 2.6. The sum and length of a code sequence have the same parity.

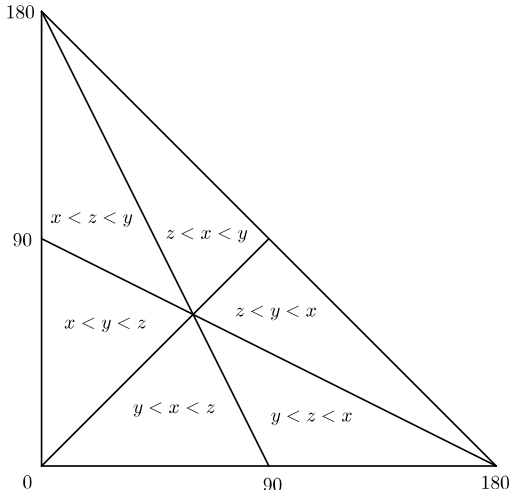
Proof. It suffices to show that a code sequence always has an even number of even numbers. From the automaton in Figure 3, this is immediate, since the corresponding binary sequence must cross between the upper and lower states an even number of times to return to the accepting state. □

3 The Space of Triangles

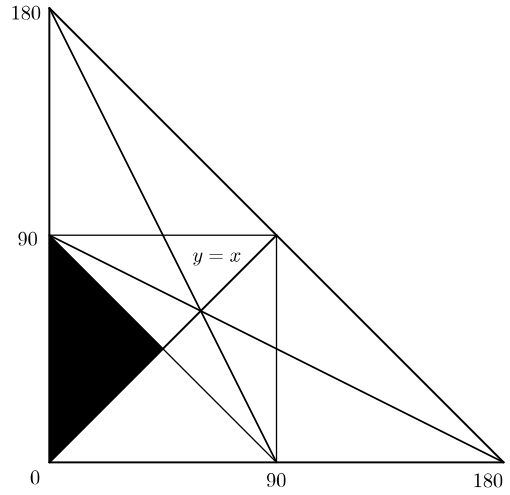
Suppose we have a triangle with angles x, y, z. Using that $z = \pi - x - y$, we can identify each triangle by the pair of angles (x, y). We will suppose that $m < A = x, m < B = y$ and $m < C = z$ and since $z = 180 - x - y$, the angles of the triangle are completely determined by the coordinates (x, y) and we will plot this point in an $X - Y$ coordinate system *only if that triangle has a periodic path*. Note this is independent of the size of the triangle. Observe that if the triangle corresponding to (a, b) has a periodic path, then so do the triangles corresponding to $(a, 180 - a - b)$, (b, a), $(b, 180 - a - b)$, $(180 - a - b, a)$ and $(180 - a - b, b)$. To prove that every triangle has a periodic path amounts to plotting every point (x, y) with $x, y > 0$ and $x + y < 180$ or equivalently to plotting every point in the region $0 < x \leq y \leq z$ as in Figure 4c.



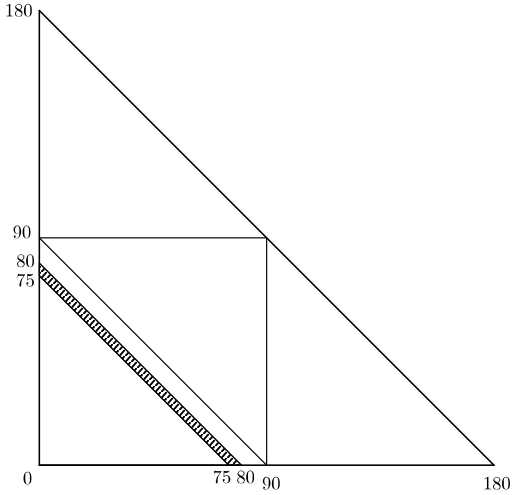
(a) All Unique Obtuse Triangles



(b) The 105 Degree Theorem



(c) All Unique Obtuse Triangles



(d) The 105 Degree Theorem

Figure 4: The Map of all Triangles

To prove that all obtuse triangles have a periodic path amounts to plotting every point (x, y) with

$x + y < 90$ and $x \leq y$ which means it is enough to fill in the shaded region of Figure 4c. Our goal in this paper is to shade in the region bounded by $x = 0$, $x = y$, $x + y = 80$ and $x + y = 67.7$ similar to the region shown in Figure 4d, or equivalently the region bounded by $x = 0$, $x \leq y$, $x + y = 80$ and $x + y = 67.7$ which in conjunction with the other known results will prove the following theorem,

Theorem 3.1 (One Hundred and Twelve Point Three Degree Theorem)**.** Every triangle whose largest angle is 112.3 degrees or less has a periodic path.

4 Unfoldings

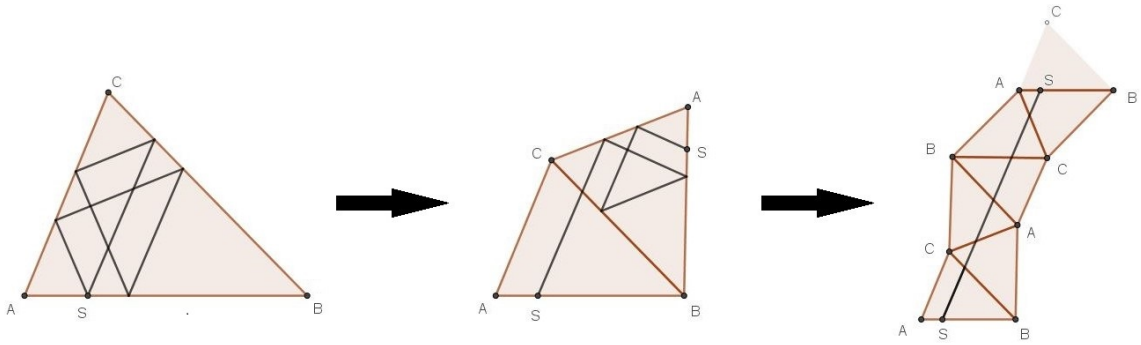


Figure 5: An example of unfolding

The methods described here will rely on another concept called an unfolding of a periodic path. These are created by instead of having the ray reflect off of a side, have the triangle itself reflect in that side and let the ray go straight through (Figure 5).

Notice that in every triangle at the end of an unfolding, there are only two directions we could possibly move, as we cannot go backwards. We can consider the two directions as left and right moves with respect to this triangle. Unfoldings can be used to look for paths more generally, as we do not need to consider any specific ray when creating an unfolding, we just need to try going both left and right at each step. The only problem is that we need a way to know when an unfolding is a periodic path. We define the side sum of an unfolding as follows: start with a side sum of 0. Then, for every triangle in the unfolding, if we will move to the right from that triangle, add the angle at the base of the reflection to the side sum. If left, subtract that angle. For example, the side sum in Figure 6 is calculated by starting at the base triangle and working up:

$$-A - A + B + B + B - C + A + A + A - B - B - B + C - A = 0 \tag{1}$$

Note that the last triangle is excluded from the sum; indeed, we will be moving neither left nor right from this triangle, so it does not make sense to add or subtract either angle, in this case the A angle or the B angle. By definition, the side sum is always a linear combination of the 3 angles in the triangle.

Next we define the orientation of triangles in the unfolding. The orientation describes how the triangle is rotated and reflected with respect to the base triangle. If the path enters through the side AB, we say the orientation is 1. If it enters though BA, it is -1. Entering though BC, CA, CB, or AC gives 2, 3, -2, and -3 respectively. For example, in Figure 6, the base triangle has an orientation of 1, and the next triangle will have an orientation of -3. The last triangle in this diagram also has orientation 1. Note that the numerical value does not have a significance, it is merely a way to identify the different states. Using these concepts, we can formulate a theorem which checks whether a periodic path is an unfolding.

Lemma 4.1. Given an unfolding of a triangle, if the side sum is 0, then the bottom of the final triangle is parallel to the bottom of the base triangle.

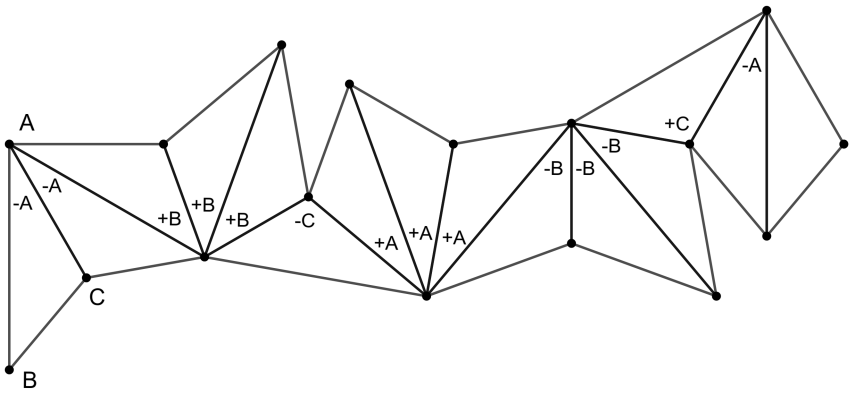


Figure 6: A Trajectory

Proof. Suppose at a certain $\triangle ABC$ in the unfolding, the bottom AB of this triangle is at an angle θ from the bottom of the base triangle. Now when we unfold, the new base will either be AC if we move left, and BC if we move right. Thus if we move left, the new bottom will be at an angle $\theta - \angle A$, and if we move right, the new angle will be $\theta + \angle B$. Thus we see that the angles of the bottoms of triangles with respect to the base triangle will correspond to the side sum at that triangle. So, if the side sum of an unfolding is 0, the base and final triangles must have an angle of 0 between them, thus they are parallel. □

Lemma 4.2. Given an unfolding of a triangle, if the side sum is 0, and the final triangle has orientation 1, and corresponding points on the edges AB and $A'B'$ of the base and final triangles can be joined by a line segment which stays inside of the unfolding, then the triangle has a periodic path given by folding up that line segment.

Proof. Since the orientation of the final triangle is 1, we know that this line segment starts on AB and ends on $A'B'$ at the corresponding point, so when folded up it will start and end at the same point. By Lemma 4.1, AB is parallel to $A'B'$, so the line segment must approach the corresponding point at the same angle from which it left, so this line segment is an unfolded periodic path. □

This provides a foundation for the proof that a path is periodic from its unfolding. The only problem is that we now need a method for checking the tricky condition that corresponding points on the base and final triangles can be joined by a line segment which stays inside the unfolding.

Definition 4.3. Given an unfolding of $\triangle ABC$, call the final triangle $A'B'C'$. We call the vector which joins the points A and A' the *shooting vector*.

Observe that if sides AB and $A'B'$ are parallel, the vectors from A to B and A' to B' are identical. We see then that in order for the poolshot to pass through corresponding points on $\triangle ABC$ and $\triangle A'B'C'$, it must be parallel to the shooting vector. Thus, in order to check that such a line can pass through the interior of the unfolding, all that is needed is that some translation of this line intersects the base of every triangle. To do this, we simply calculate every vector (c, d) from a point on the left side of the unfolding to a point on the right, and check that $ad - bc < 0$.

Using this observation along with Lemma 4.2, we now have a theorem that is much clearer, and most importantly, easy for a computer to check.

Theorem 4.4 (Basic Periodic Path Test)**.** An unfolding of a triangle is periodic if and only if the side sum is 0, the final triangle has orientation 1, and the shooting angle $w = (a, b)$ has the property that $ad - bc < 0$ for all vectors v where $v = (c, d)$ is a non-zero vector from any blue point to any black point.

While this theorem is workable, there is one key optimization that can be made utilizing the code sequence notation, namely *fans*.

4.1 Fans

Definition 4.5. A *fan* is a tower of mirror images of a triangle in which all successive mirror images alternate between the same two sides which means that all triangles of the fan intersect at the same vertex which we will call its *center*. We can further classify the fans as *blue fans* if the center is black and all other vertices are blue points or *black fans* if its center is blue and all other vertices are black. The *central angle* of a blue or black fan is the maximal angle at its center produced by the blue or black vertices in the tower.

Any tower can be viewed as a succession of fans and in particular any poolshot tower can be viewed as a succession of alternating black and blue fans as cut off by the straightened poolshot. The blue points of a blue fan lie on one or two circular arcs whose endpoints are vertices of the fan. The endpoints of those arcs are called the key points. Each fan has either 2, 3 or 4 key points. In case if one of the arcs is a single point, there are 3 key points. Otherwise there are 2 or 4. Similar for the black points.

Observe that the center of a fan is a key point of the following and preceding fan of opposite color, assuming it has a following or preceding fan.

Given a blue fan in a poolshot tower, then the key points of each blue arc are the points on the arc closest to the straightened poolshot. Similarly for black fans. This is a consequence of the fact that $\sin\theta$ is a minimum at the endpoints of the interval $[a, b]$ where $0 < a \leq b < 180$ or equivalently that $\cos\theta$ is a minimum at the endpoints of the interval $[a, b]$ where $-90 < a \leq b < 90$ and that in a poolshot tower the central angle of any fan is less than 180 degrees.

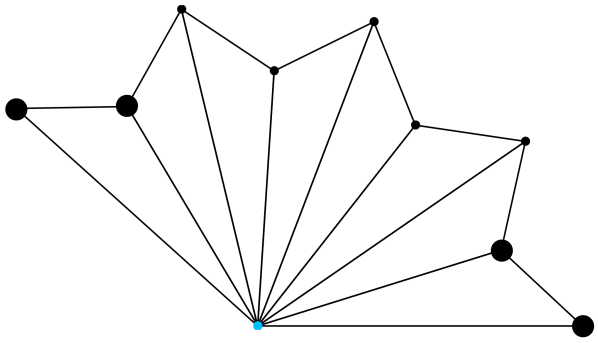


Figure 7: A black fan with 4 key points

Theorem 4.6 (Improved Periodic Path Test)**.** Consider an unfolding of a triangle where the central angle of any blue or black fan is less than 180 degrees. This unfolding is a periodic path if and only if the side sum is 0, the final triangle has orientation 1, and the shooting angle $w = (a, b)$ has the property that $ad - bc < 0$ for all vectors v where $v = (c, d)$ is a non-zero vector from any key blue point to any key black point.

Proof. Choose our coordinate system so that the line AA' is vertical. Let V be a black point with the smallest X coordinate in our system and let L_2 be a vertical line through V. Similarly, let U be a blue point with the largest X coordinate and let L_1 be a vertical line through U. Then our tower is a periodic poolshot tower if and only if U lies to the strict left of V.

If the tower is a periodic poolshot tower then by 4.4, $ad - bc < 0$ for all vectors from blue points to black points and hence this is certainly true for all vectors from key blue points to key black points.

To show the other direction, it is enough to show that U and V are both key points since then the condition that $ad - bc < 0$ will guarantee that U lies to the strict left of V. Now since V is black, it must lie on some black arc whose center is blue. If that blue center lies to the left of or on L_2, then V must be a key black point otherwise since the central angle is less than 180 degrees one of the black endpoints of the black arc through V would lie to the left of V. If that blue center lies to the right of L_2 , then since that center is a key blue point and the endpoints of the black arc through V are key black points, the condition $ad - bc < 0$ would force the central angle of that black arc to be greater than 180 degrees which is impossible. Hence this case cant arise and V is a key black point. Similarly U is a key blue point. □

4.2 The XYZ Algorithm Rules

In our periodic path tests, we need to be able to calculate the side sum of a code sequence, as well as find the shooting angle which would give the poolshot vector. To do this more efficiently and clearly than in (1), we make use of the fact that the numbers of a code sequence represent the sizes of the fans in that unfolding. What is required then is an algorithm which will give the angle in each fan corresponding to each code number.

Algorithm 4.1 The XYZ algorithm

```
1: function XYZ(sequence)
2:     prev, curr := x, y                ▷ The angles here could be chosen to be any of x y or z
3:     xyzequation := ""
4:     for num in sequence
5:         next := if num % 2 = 0 then prev else OtherAngle(prev, curr) (from Algorithm 2.1)
6:         prev, curr := curr, next
7:         Append num + Uppercase(curr) + " " to xyzequation
8:     end for
9:     return xyzequation
10: end function
```

For example, applying this algorithm to the code sequence 1 3 3 1 3 3 shows that the alternating central angles of the fans are $1X\ 3Y\ 3Z\ 1X\ 3Y\ 3Z$ in that order. This is what we call the XYZ equation, and we say that the items alternate between being top or bottom angles, as a way to identify the alternating directions of the fans. The visual representation of this process is:

$$\begin{array}{cccccc} X & & Z & & Y & \\ 1 & 3 & 3 & 1 & 3 & 3 \\ & Y & & X & & Z \end{array} \tag{2}$$

Given a periodic poolshot tower, we call the successive acute angles as the poolshot crosses each side the shooting angles. If the first shooting angle from the base of the tower is θ where $0 < \theta < 90$, the first fan it crosses involves the angle X with central angle nX, and if $n = 2k + 1$ or $n = 2k$, then the successive shooting angles are, respectively,

$$\begin{array}{ll} \theta, & \theta, \\ \theta + X, & \theta + X, \\ \theta + 2X, & \theta + 2X, \\ \ldots & \ldots \\ \theta + (k-1)X, & \theta + (k-1)X, \\ \theta + kX, & \theta + kX \ // \ 180 - \theta - kX, \\ 180 - \theta - (k+1)X, & 180 - \theta - (k+1)X, \\ 180 - \theta - (k+2)X, & 180 - \theta - (k+2)X, \\ \ldots & \ldots \\ 180 - \theta - (2k+1)X & 180 - \theta - 2kX \end{array} \tag{3}$$

where the // indicates that either $\theta + kx$ or $180 - \theta - kx$ may be the acute angle at this stage. Note that the first shooting angle cannot be 90 since in any fan that contains the 90 degree shooting angle, the side with the 90 degree shooting angle is also right in the center of the fan. Also observe that all the angles are integer linear combinations of x, y, 180 and θ and that as the poolshot passes from one fan to the next all the shooting angles are completely determined. In particular, if θ can be expressed as an integer linear combination of x, y and 90 then so too can every shooting angle.

4.3 Calculating the Coordinates of the Vertices in a Code Tower

The next thing in our periodic path tests is that we need to be able to find the locations of each key vertex in the tower. First, we must give the method we use to label the vertices in the tower.

We will assume that the base is A_0B_0 and the first reflection is in side A_0C_0, then the black point B_0 has the label $L_{(1,0)}$ and the blue point A_0 the label $L_{(2,0)}$. Now as the centers alternate between black and blue points the labels increase by one. Observe that all black centers have odd labels $L_{(2i-1,0)}$ and all blue centers have even labels $L_{(2i,0)}$ for $i \geq 1$. If the tower has $2m$ fans in it, then the last labels are $L_{(2m+2,0)}$ and $L_{(2m+1,0)}$ which belong to the last A and B vertices in the tower respectively.

As to the key points if any belonging to the fan with center $L_{(k,0)}$, if there are four of them the first one appearing in the tower after $L_{(k-1,0)}$ is labeled $L_{(k,1)}$ and the second $L_{(k,2)}$ whereas if there are three key points, the one after $L_{(k-1,0)}$ is labeled $L_{(k,1)}$.

Observe that the number of fans in a periodic poolshot tower in standard form equals the number of code numbers which is always even where we remind the reader that periodic paths of odd side sequence length are doubled in order to get a parallel tower.

The first B and the last A vertices can be considered as centers of degenerate fans involving no triangles and are key points and are not counted in the number of fans.

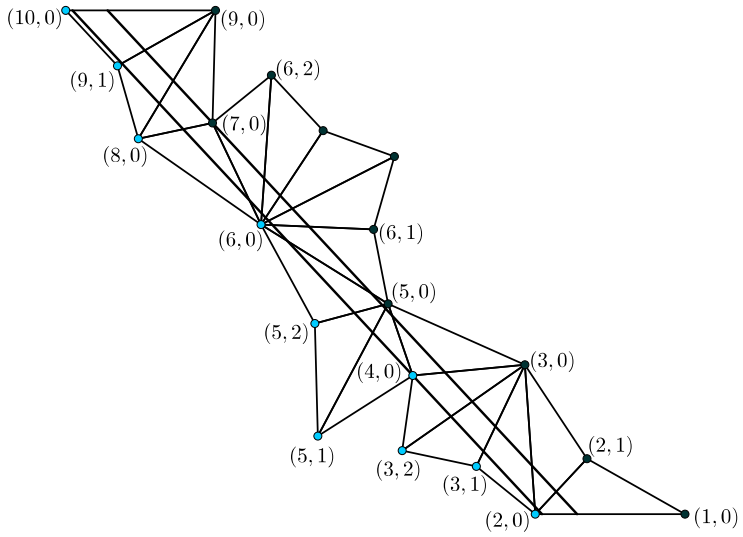


Figure 8: Labelled Periodic Poolshot Tower

Given a code tower of even length, assume that the ordering of the angles in the code tower is such that the first top angle is $U_1 = X$ and that the last bottom angle is $U_{2k} = Z$ as shown:

$$\begin{array}{ccccccc} U_1 & & U_3 & & \\ C_1 & C_2 & C_3 & \ldots & C_{2k} \\ & U_2 & & & U_{2k} \end{array} \tag{4}$$

Then the base of the tower is AB with $\angle A = x$, $\angle B = z$, and $\angle C = y$. If we let $AB = \sin y$, then $BC = \sin x$ and $AC = \sin z = \sin(x + y)$ since $z = 180 - x - y$ we then can recursively calculate the coordinates of each fan center $L(i, 0)$ as follows: Letting $a_i = U_iC_i$ be the center angle of the fan with corresponding label $L_{(i+1,0)}$ for $i \geq 1$, u_i be the length of the side between (x_i, y_i) and (x_{i+1}, y_{i+1}) where (x_i, y_i) are the coordinates of the center of the fan at $L_{(i,0)}$ then recursively let $x_1 = \sin y$, $y_1 = 0$, $x_2 = 0$, $y_2 = 0$ and proceed:

$$
\begin{aligned}
x_{2n} &= x_{2n-1} - u_{2n-1}\cos(a_{2n-2} - a_{2n-3} + a_{2n-4}... - a_3 + a_2 - a_1) \\
y_{2n} &= y_{2n-1} - u_{2n-1}\sin(a_{2n-2} - a_{2n-3} + a_{2n-4}... - a_3 + a_2 - a_1) \\
x_{2n+1} &= x_{2n} - u_{2n}\cos(a_{2n-1} - a_{2n-2} + a_{2n-3}... + a_3 - a_2 + a_1) \\
y_{2n+1} &= y_{2n} - u_{2n}\cos(a_{2n-1} - a_{2n-2} + a_{2n-3}... + a_3 - a_2 + a_1)
\end{aligned}
$$

An example of this process is worked through in the appendix A.

5 Classifying Codes

5.1 Stable and Unstable

Definition 5.1. A code sequence is called *stable* if the sum of all the top angles involving X times its code number equals the sum of all the bottom angles involving X times its code number, and similarly for all the angles involving Y and all the angles involving Z. Otherwise, it is called *unstable*.

There are stable codes for which there are no triangles (x, y) which have a periodic path corresponding to that code sequence. We will make the convention that we will only call a code sequence stable if there is at least one triangle (x, y) which has a periodic path corresponding to that code sequence. If that is the case, then since the code is stable there is a finite periodic path within the triangle whose straightened trajectory is a positive minimum distance from all vertices. This means we can always change the coordinates (x, y) by a small amount and have another different triangle with the same periodic path. The upshot of this is that a region corresponding to a stable code is an open non-empty set in the plane and would cover a positive area. We will call it the *tile* or the *region* corresponding to that stable code sequence.

Definition 5.2. We will call a code sequence *unstable* if it is not stable and the sum of the top angles multiplied by their code numbers equals the sum of the bottom angles multiplied by their code numbers.

Once again, we will not consider such a code to be unstable if there do not exist any triangles (x, y) that have a periodic path corresponding to that code sequence. One example: 1 2 1 2 is an unstable code sequence since along the line $x + y = 90$ there is a periodic path of type 1 2 1 2. In fact it can be easily shown that every right triangle has a periodic path of this type.

Observe that an unstable code sequence corresponds to a finite open line segment whose equation is determined as in the previous section.

Intuitively, a stable path is one whose sum is always 0 no matter what the X, Y, and Z angles are, and an unstable path is a path whose sum is 0 only for some specific triangles. From this, you can see why slightly changing the triangle will cause an unstable path to fail, where a stable path may still work.

5.2 Even and Odd

Definition 5.3. If a code's length is even, we call it an *even* code. Otherwise, we call it an *odd* code.

If a periodic poolshot is of odd length and finishes at $A'B'$, then $A'B'$ is antiparallel (in the sense that interior angles on the same side of the straightened poolshot are equal) to AB with A a blue point and A' a black point. It follows that our poolshot tower will have a start and end which are parallel if and only if we double the length of the code when creating the tower. This results in the code sequences representing odd paths needing to be doubled when applying the periodic path tests and calculating the side sums.

Lemma 5.4. All odd codes are stable.

Proof. Because we calculate an odd code by doubling it's length, it is clear that any valid odd code will have the same number of Xs, Ys and Zs on top and bottom and thus will be stable. □

5.3 Perpendicular

Definition 5.5. If the angle sequence of a code can be rotated such that it becomes a palindrome, it is called *perpendicular*. Otherwise, it is called *oblique*.

An equivalent definition for perpendicular paths is that that the periodic poolshot in the code tower will strike a side at 90 degrees in two places. Intuitively that is because the palindromic code happens when the path traces over the same triangles once forwards and backwards, similiar to what would happen if the poolshot struck a side perpendicularly.

Perpendicular paths are extremely nice to work with, as we know that taking a shooting angle of 90 degrees is guaranteed to work. This leads us to an even further improved version of 4.6 which works only for perpendicular paths:

Theorem 5.6 (Perpendicular Periodic Path Test)**.** Given a perpendicular code sequence, permute the tower such that its side sequence is a palindrome. Then the tower is a periodic poolshot tower if and only if $ad - bc < 0$ where $v = (c, d)$ is a non-zero vector from any key blue point to any key black point which lie between or on the triangles in the first half of the tower, and $w = (a, b)$ is the poolshot vector provided the angle of any blue or black fan is less than 180 degrees.

From the definition one may expect that the perpendicular codes are of even length, but there is actually a stronger result.

Proposition 5.7 ([7])**.** The period of all perpendicular codes is a multiple of 4.

Corollary 5.8. All perpendicular codes are even. Equivalently, no odd codes are perpendicular.

5.4 The Five Code Types

As there are three properties we use to divide the paths, one would expect eight classes. Lemma 5.4 and Prop 5.8 tell us that all odd codes are stable and oblique, thus we are left with only five classes.

Using the periodic tests in Theorems 4.6 and 5.6 requires the use of the key points and the shooting vector. The use of this classification is that each type of path has different properties, which in some cases can result in shortcuts being available in computation of the shooting vector.

1. OSO: odd, stable, oblique. The first example of such a code is 1 1 1, which is the code corresponding to the orthic triangle in an acute triangle.

   Finding the poolshot vector for these paths is quite simple. Since the code will be of the form $A_1A_2...A_{2n-1}$, doing the XYZ algorithm with a shooting angle of θ will give the final angle as

   $$B_0 \cdot 180 - \theta \pm B_1X \pm B_2Y \pm B_3Z$$

   for $B_0 \in \{-1, 0, 1\}$ and some integers B_1, B_2, B_3. If this path were periodic, this expression is equal to θ itself, and thus we are able to solve for the required shooting angle θ. The key point here is that since there are an odd number of code numbers, the θ term alternates between positive and negative an odd number of times, making it odd in the end. As an illustration of this idea, consider the OSO code 1 1 2 2 5. Using the XYZ algorithm,

   $$\begin{array}{ccccccccc} X & & & & Z & & & & Z \\ 1 & & 2 & & 2 & & 5 & & 5 \\ & & Y & & & & Y & & \end{array} \tag{5}$$

   we get a final equation for the side sum allowing us to solve for θ using the rule in equation 3,

   $$180 - \theta - 1X + 1Y - 2Z + 2Y - 5Z = \theta$$

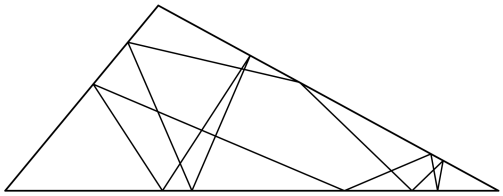


Figure 9: An OSO path, 1 1 2 5 2

2. ESO: even, stable, oblique. The first example of such a code is 1 1 2 2 1 1 3 3.

   It can be shown that the shooting angles are not in general integer or even rational linear combinations of X, Y and 90, and we will show by example why the trick for the OSO codes fails here. Because of this, the poolshot vector $(c, d) = (-\cos\theta, \sin\theta)$ used in the periodic poolshot tower test is calculated by first finding the vertex points and calculating which shooting angle would be needed to shoot from a vertex in the starting triangle to one in the final triangle, as in Appendix A. There is a way to eliminate θ to form a bounding polygon which is discussed later in section 8 and which contains the straight line segment region corresponding to the ESO code.

   For the example, consider the XYZ algorithm applied to 1 1 2 2 1 1 3 3

$$\begin{array}{cccccccc} X & & Z & & Z & & Y & \\ 1 & 1 & 2 & 2 & 1 & 1 & 3 & 3 \\ & Y & & Y & & X & & Z \end{array} \tag{6}$$

   which yields the final equation

$$\theta + 1X - 1Y + 2Z - 2Y + 1Z - 1X + 3Y - 3Z = \theta$$

   which is of course no help when trying to recover θ.

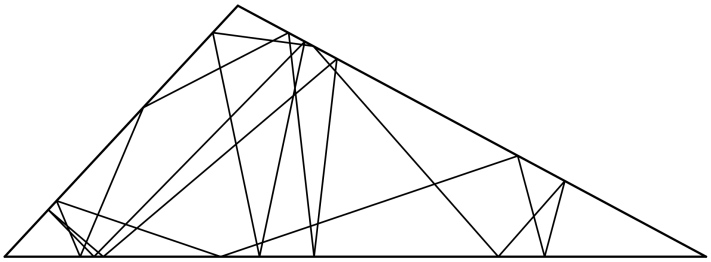


Figure 10: An ESO path, 1 1 2 2 3 1 3 5

3. ESP: even, stable, perpendicular. The first example is 1 1 1 1 2 1 1 1 1 2.

   For these codes, the shooting angle is already known to be 90 degrees for some permutation of the angle sequence, and checking which one is easy since it will be the palindromic angle sequence.

4. EUO: even, unstable, oblique. One of the first examples is 1 1 2 1 3 2.

   Similar to ESO codes, the shooting angles are not in general rational linear combinations of X, Y and 90, and so we again need to calculate the shooting angle as in Appendix A.

5. EUP: even, unstable, perpendicular. One of the first examples is 1 2 1 6.

   Just like for ESP codes, the shooting angle is already known to be 90 degrees for some permutation of the angle sequence.

6 Previous Results on Periodic Paths

We'll now give an overview of previous results on periodic paths, to show how they fit into our notation.

1. Acute triangles always have the OSO periodic path 1 1 1. This path corresponds to the orthic triangle, whose vertices are the feet of the altitudes. This result is due to Fagnano.

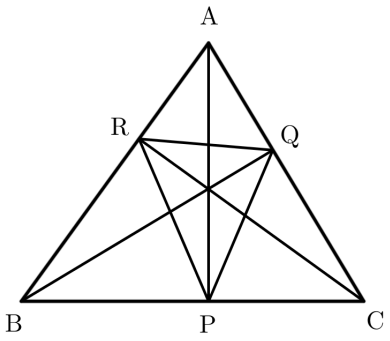


Figure 11: Orthic Triangle

2. Right triangles always have the ESP periodic path 1 2 1 2.

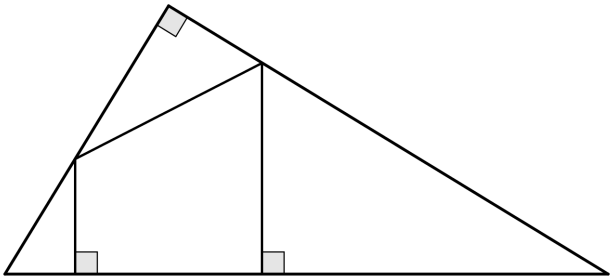


Figure 12: Right Triangle with ESP Periodic Path 1 2 1 2

3. Isosceles triangles always have the EUP perodic path 2 2.

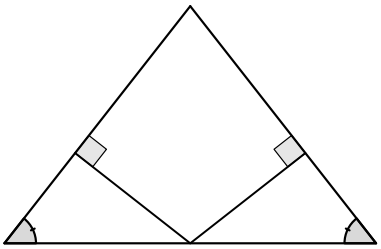


Figure 13: Isosceles triangle with EUP Periodic Path 2 2

4. Rational triangles always have a periodic path of type ESP. A triangle is *rational* if all three angles can be expressed as integer multiples of x where x divides 90, say $\angle A = nx$, $\angle B = mx$ and $\angle C = px$. As mentioned before, this result is originally due to Masur [2], but we will go though a sketch of the proof here.

Observe that if a poolshot leaves a side at 90 degrees and $90 = qx$ for some integer q, then it bounces off any side at some integer multiple of x. This implies that there are at most q angles involved as a poolshot bounces off the sides of the triangle.

Further, If a poolshot leaves a side at 90 degrees and $90 = qx$ for some integer x and hits a vertex say vertex B, then it must enter B along a ray making an angle tx where $0 < t < m$.

We conclude that there are at most $n + m + p - 3$ perpendicular poolshots which hit a vertex in a rational triangle. This means if a 90 degree poolshot leaves a side and doesn't hit a vertex, there is an open band around that poolshot of finite width $\delta > 0$ which leaves that side at 90 and never hits a vertex and in which δ is maximal. This is only possible if the band hits another side at 90 and becomes periodic since otherwise the band must hit some side, say side AB, infinitely often at some angle jx for a fixed integer j less than q and which is repeated infinitely often. But each time this band hits this side, it hits on some open interval (a_i, b_i) of width δ no two of which can intersect without contradicting the maximality of δ. Since AB is finite, this is impossible.

7 The Two Infinite Patterns

There are two infinite patterns which converge to the line segment $x = 0$, $67.5 < y < 90$. The first pattern is the same as the one that is used in [3], but the second is new and uses thin regions which are unstable. As we will show in a moment, these two patterns tessellate a cover although they do not overlap. Notice that pattern I is an open region, while pattern II is made up of unstable codes and thus creates line segments which are open at the endpoints. This contrast this with [3] which uses two stable overlapping patterns.

Proposition 7.1 (Infinite Pattern I)**.** Any triangle ABC with $m < A = x$, $m < B = y$ where

$$(n+1)x + 2y < 180 < (n+2)x + 2y \text{ and } 0 < x < \frac{90}{2n+2}$$

for $n \geq 1$, contains a ESP periodic path 1 1 $(2n+1)$ 1 2 1 $(2n+1)$ 1 1 $(4n+2)$.

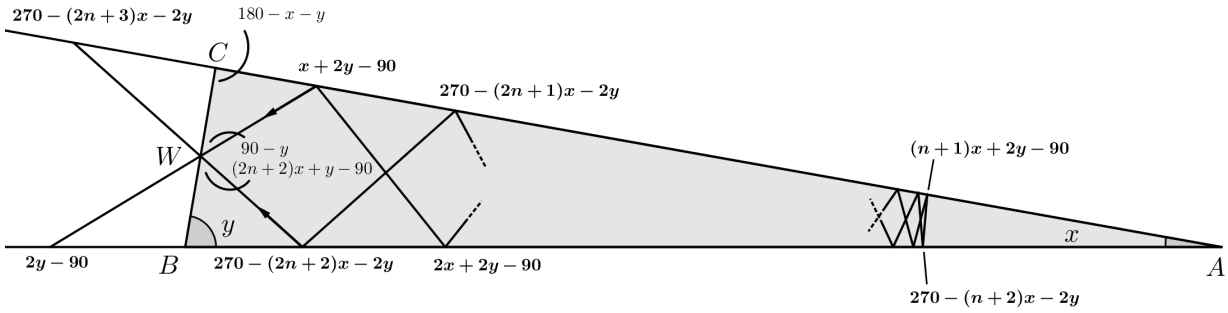
Proof. For the $n \geq 2$ even case, take a wedge of acute angle x and vertex A with arms l_1 and l_2 as shown on Figure 14a and pick a point P on l_1 and shoot a poolball at an acute angle $0 < 270 - (n+2)x - 2y < 90$ as shown. Now since $270-(n+2)x-2y+x < 180$, it hits l_2 at an angle $0 < (n+1)x+2y-90 < 90$ and continues bouncing all on the wedge at angles $nx + 2y - 90 > (n-1)x + 2y - 90 > ... > x + 2y - 90 > 2y - 90 > 0$ as shown. If we consider the ray from P in the other direction, it bounces off the sides at the angles $270-(n+3)x-2y > 270-(n+4)x-2y > ... > 270-(2n+3)x-2y > 0$ where the last inequality holds since $(n+1)x + 2y < 180$ and $(n+2)x < (2n+2)x < 90$. Now let W be the intersection of the last two rays and draw a line through W hitting l_1 at B, l_2 at C and such that the angle at B as shown is y. Observe that B lies between the last two reflections on l_1 since $y > 2y - 90$ since $y > 45$ and $180 - y > 270 - (2n+2)x - 2y$ since $(2n+2)x + y > 90$ where this last inequality holds since $(2n+2)x + y > (n+2)x/2 + y > 90$. The last ray from l_2 hits BC at W at the angle $90 - y$ and bounces to hit AB at 90 whereas the last ray from l_1 hits W at $(2n+2)x + y - 90$ and since $(2n+2)x + y - 90 + 180 - x - y = 90 + (2n+1)x < 90 + (2n+2)x < 180$, it reflects off BC and hits l_2 at $90 - (2n+1)x$. Observing that $90 - (2n+1)x > x$ since $(2n+2)x < 90$, the ray then bounces off l_1 and l_2 until it hits at 90 producing a ESP periodic path 1 1 $2n+1$ 1 2 1 $2n+1$ 1 1 $4n+2$.

The odd case is handled similarly interchanging l_1 and l_2.

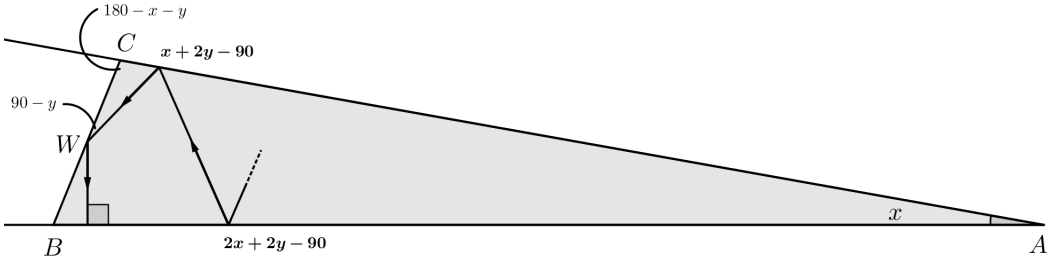
□

Remark 7.2. Since $x < 22.5$ and $y + \left(\frac{n+2}{2}\right) x > 90$ and $\left(\frac{n+2}{2}\right) x < \left(\frac{n+2}{2n+2}\right) 22.5 < 22.5$ then $y > 67.5$. Also observe that since $n \geq 1$, then $2x + 2y \leq (n+1)x + 2y < 180$ which means that $x + y < 90$. Finally observe that the successive regions determined by these conditions share the boundary line $(n+1)x + 2y = 180$ for $n \geq 2$. In particular, the segments in the following proposition are precisely the boundary lines (extended) between the those regions.

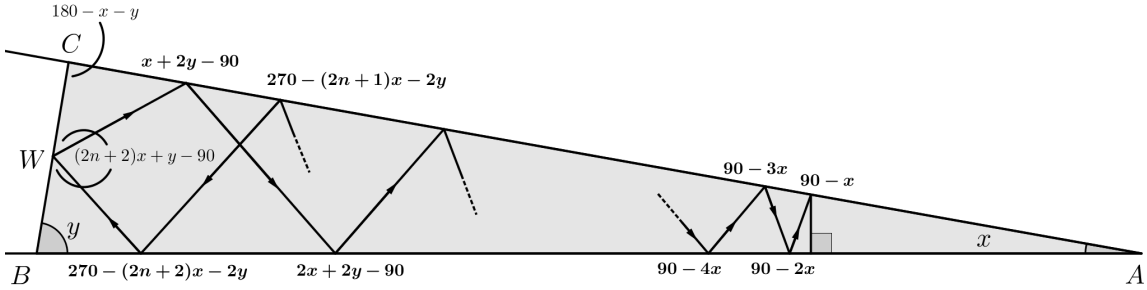
Proposition 7.3 (Infinite Pattern II)**.** Any triangle ABC with $m < A = x$, $m < B = y$ where $(n+1)x+2y = 180$ and $0 < x < 90/n$ for $n \geq 1$ contains a EUP periodic path 1 2 1 $2n$.



(a) $n \geq 2$ even case



(b) $n \geq 2$ even case



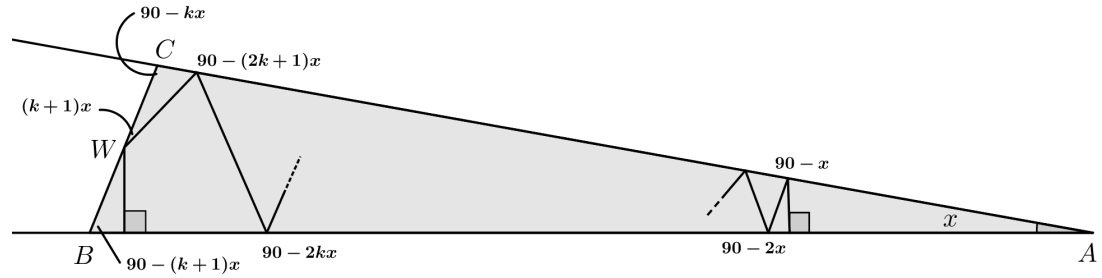
(c) $n \geq 2$ even case

Figure 14: Infinite Pattern I

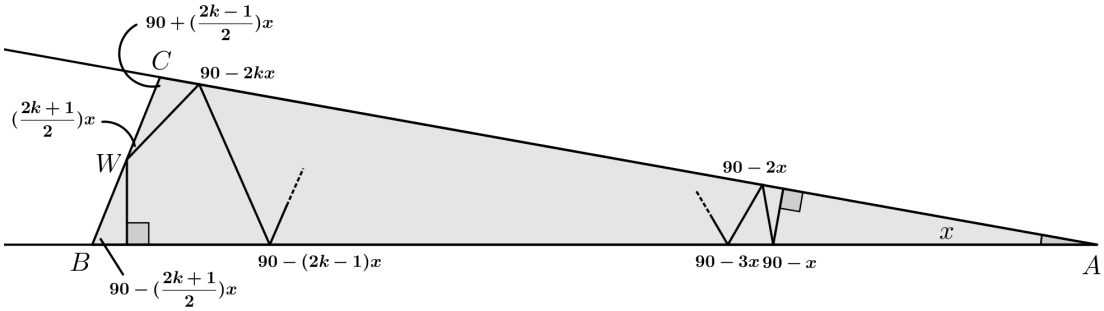
Proof. For the odd integer case $n = 2k+1$, $k \geq 0$, take a wedge of angle x and vertex A with arms l_1 and l_2 as shown in Figure 15a and shoot a poolball at 90 degrees from l_1 which then bounces off the sides at angles $90 - x > 90 - 2x > ... > 90 - (2k+1)x > 0$ noting that $(2k+1)x < 90$. Now on the last ray leaving l_2 at angle $90-(2k+1)x$, choose any point W between l_1 and l_2 and draw a line through W hitting l_1 at B at an acute angle $y = 90 - (k+1)x > 0$ (and so $(n+1)x + 2y = 180$) and hitting l_2 at C. Observe that triangle ABC has a periodic path of type 1 2 1 $2n$.

For the even integer case $n = 2k$, $k \geq 1$, again take a wedge of angle x and vertex A with arms l_1 and l_2 as shown in Figure 15b and shoot a poolball at 90 degrees from l_2 which then bounces off the sides at angles $90 - x > 90 - 2x > ... > 90 - 2kx > 0$ noting that $2kx < 90$. On the last ray leaving l_2 at angle $90-2kx$, choose any point W between l_1 and l_2 and draw a line through W hitting l_1 at B at an acute angle $y = 90 - (2k+1)x/2 > 0$ (and so $(n+1)x + 2y = 180$) and hitting l_2 at C. Observe that triangle ABC has a periodic path of type 1 2 1 $2n$. □

Given an obtuse triangle ABC, with$0 < x < 22.5$, $67.5 < y < 90$ and $x + y < 90$, then that triangle has a periodic path. This can also be seen in [3] using a different pair of infinite patterns.



(a) Infinite Pattern 2 Even Case



(b) Infinite Pattern 2 Odd Case

Figure 15: Infinite Pattern II

8 The Prover

The basic idea behind the prover is the Mean Value Theorem in two dimensions from calculus.

Given a code sequence and its corresponding code tile R, we would like to verify that the code sequence represents a periodic poolshot at every point (x, y) in R. It is enough to show that the following functions of two variables are positive on a small enough square to be completely contained in R. We then combine squares and even different code sequences to create a cover as in Figure 18. Due to the massive data set, we made no attempt to optimize the cover. That is, any time found a code that was proven to cover an uncovered point, we added it to the cover.

Theorem 8.1 (Mean Value Theorem)**.** Let $f(x, y)$ be a differentiable function. Then on the line between any two points (a_1, a_2) and (b_1, b_2), there is a point (c_1, c_2) such that

$$f(b_1, b_2) - f(a_1, a_2) = f_x(c_1, c_2)(b_1 - a_1) + f_y(c_1, c_2)(b_2 - a_2)$$

Observe that if $f(x, y) = \sum_{i=1}^{k} \pm u_i \cos(m_i x + n_i y)$, then $f_x = \sum_{i=1}^{k} \mp m_i u_i \sin(m - ix = n_i y)$ and $f_y = \sum_{i=1}^{k} \mp n_i u_i \sin(m - ix = n_i y)$, and hence $|f_x| \leq \sum_{i=1}^{k} |m_i u_i| =: M$, and $|f_y| \leq \sum_{i=1}^{k} |n_i u_i| =: N$. This holds for the same M and N if $f(x, y) = \sum_{i=1}^{k} \pm u_i \sin(m_i x + n_i y)$.

If $f(b_1, b_2) > 0$ and $f(a_1, a_2) \leq 0$, then by the mean value theorem we have

$$f(b_1, b_2) - f(a_1, a_2) = f_x(c_1, c_2)(b_1 - a_1) + f_y(c_1, c_2)(b_2 - a_2) \geq f(b_1, b_2)$$

and since

$$f_x(c_1, c_2)(b_1 - a_1) + f_y(c_1, c_2)(b_2 - a_2) \leq M\,|b_1 - a_1| + N\,|b_2 - a_2|$$

we have

$$f(b_1, b_2) \leq M\,|b_1 - a_1| + N\,|b_2 - a_2|$$

In particular, this means that if $f(b_1, b_2) > M\,|b_1 - a_1| + N\,|b_2 - a_2|$, then $f(a_1, a_2) > 0$.

Now, let (b_1, b_2) be the center of a square of side length $2r$. Then for any (a_1, a_2) in or on the boundary of the square, we must have $(b_1 - a_1), (b_2 - a_2) \leq r$, and thus $M\,|b_1 - a_1| + N\,|b_2 - a_2| \leq (M + N)r$.

This means that if

$$f(b_1, b_2) > (M + N)r \geq M\,|b_1 - a_1| + N\,|b_2 - a_2|$$

then $f(a_1, a_2) > 0$. Even further, if $0 < r < f(b_1, b_2)/(M + N)$ and $f(b_1, b_2) > 0$, then all points (a_1, a_2) in the square centered at (b_1, b_2) satisfy $f(a_1, a_2) > 0$. This leads us to our algorithm for the prover, using the fact from Section 4.3 that the locations of the vertices can be written as such functions of sin and cos.

Theorem 8.2 (The Gradient Algorithm)**.** For every set forming the boundary equations a periodic path, consisting functions of the form $f_j(x, y) = \sum_{i=1}^{k} \pm u_i \cos(m_i x + n_i y)$ or $f_j(x, y) = \sum_{i=1}^{k} \pm u_i \sin(m_i x + n_i y)$, the following scheme verifies that the path holds over a square of side length $2r$ centered at (b_1, b_2).

1. Compute $G_j = \sum_{i=1}^{k} |u_i|\,(|m_i| + |n_i|)$
2. Verify that $f_j(b_1, b_2) - rG_j > 0$ for all f_j.

8.1 Triple Rule Algorithm

Suppose two code regions R_1 and R_2 intersect along a common boundary line segment from (x_1, y_1) to (x_2, y_2). This can happen if R_1 is defined by the equations $f_i(x, y) > 0$ and $f(x, y) > 0$ and R_2 is defined by the equations $g_i(x, y) > 0$ and $g(x, y) > 0$ and the equations $f(x, y) = 0$ and $g(x, y) = 0$ have a common factor of the form $\sin(ax+by)$ or $\cos(ax+by)$. Now observe that $\sin(ax+by) = 0$ if and only if $ax+by = 180k$ and $\cos(ax + by) = 0$ if and only if $ax + by = 90 + 180k$ for some integer k.

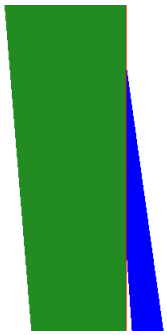


Figure 16: Two Stable Regions Sharing a Line Segment

Let us illustrate with the sine case and let us further suppose that R_1 lies between the parallel lines $ax + by = 180(k - 1)$ and $ax + by = 180k$ and that R_2 lies between the parallel lines $ax + by = 180k$ and $ax + by = 180(k + 1)$ and that $\sin(ax + by) > 0$ between the first two parallels and $\sin(ax + by) < 0$ between the second two parallels. Let $f(x, y) = \sin(ax + by)u(x, y)$ and $g(x, y) = \sin(ax + by)v(x, y)$

Now consider a square with sides parallel to the coordinate axis whose vertex coordinates are all rational numbers and which lies between the first and third parallels and which may or may not intersect the second parallel. It is worth noting that if this square lies inside either code region then it cannot intersect any of the three parallel lines above since $\sin(ax + by)$ is zero there. We can use the following to decide if every point in the square including its boundary has a periodic path.

Theorem 8.3 (Triple Rule Algorithm)**.** With interval arithmetic (see section 8.3), if we can show that:

1. Each of the four corners (x, y) of the square satisfy $180(k + 1) > ax + by > 180(k - 1)$
2. The center of the square (x_0, y_0) satisfies $f_i(x, y) > 0$, $g_i(x, y) > 0$, $u(x, y) > 0$ and $-v(x, y) > 0$ (noting that we use $-v$ since we assume $\sin(ax+by) < 0$ between the second pair of parallels) and each of these equations satisfies the Gradient Algorithm with respect to the given square. It then follows that all points on the square and its boundary satisfy these inequalities.

3. Each point on the common boundary line segment $ax + by = 180k$ from (x_1, y_1) to (x_2, y_2) has a periodic path corresponding to a EUP or EUO code which includes this line segment and runs from from (x_3, y_3) to (x_4, y_4) and that each of the four corners of the given square lies between the lines $x = x_3$ and $x = x_4$ or between the lines $y = y_3$ and $y = y_4$.

Then that square must lie within R_1 union R_2 union R_3 where R_3 is the linear region corresponding to the EUP or EUO code from 3 and every point in that union has a periodic path.

Proof. If all points on the square satisfy $\sin(ax+by) > 0$, it is within R_1. If all points satisfy $\sin(ax+by) < 0$, it is within R_2. Otherwise it is within R_1 union R_2 union R_3. □

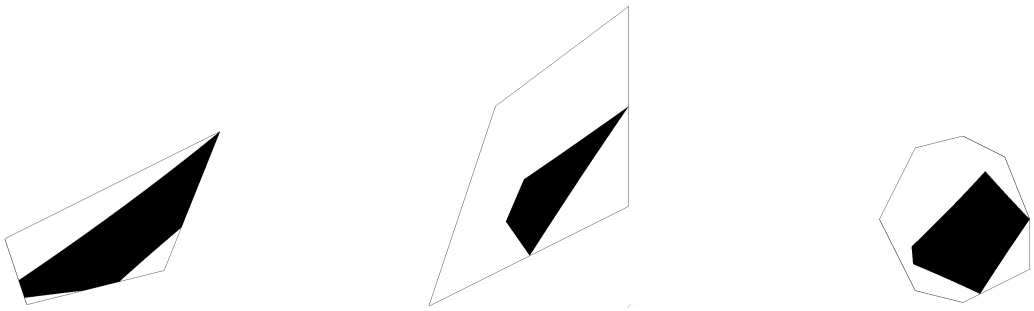
8.2 Bounding Polygons

Definition 8.4. A *bounding polygon* is a convex polygon in which a code region lies inside.

Every code region has a bounding polygon, for example the region bounded by $0 < x + y < 180$. It is useful in our calculations to find a bounding polygon with rational vertices which is as small as possible, the smallest being the convex hull of the region. There are up to six bounding polygons for each code, one corresponding to each permutation of the angles. For each code, we will assume we have a fixed order of the code angles X, Y and Z.

Definition 8.5. The *corner bounding polygon* is the polygon determined by the conditions that $0 < nX < 180, 0 < mY < 180$ and $0 < pZ < 180$ where nX, mY, pZ are the code angles corresponding to the largest X, Y, Z code numbers in the code sequence.

Definition 8.6. The *angle bounding polygon* is a polygon calculated as follows. Given a periodic side sequence leaving side AB of triangle ABC at an angle T where $0 < T \leq 90$, we can calculate its successive angles as it reflects off each side. Since $z = 180 - x - y$, these reflecting angles will be linear combinations of x, y, 90 and T with integer coefficients. If T can be expressed in terms of x, y and 90 with integer coefficients, then so can all reflecting angles. This is the case for the OSO, ESP and EUP periodic paths but not for the ESO or EUO periodic paths. An example is given below.



(a) ESP Code 1 3 6 3 1 7 3 1 8 1 3 7 (b) OSO Code 1 1 2 2 3 1 2 1 4 (c) ESO Code 1 1 2 2 3 1 3 5

Figure 17: Angle Bounding Polygons

Now observe that in the OSO, ESP and EUP cases each reflecting angle θ must satisfy $0 < \theta \leq 90$ and if we omit the 90 degree angles, then the set of points (x, y) satisfying $0 < \theta < 90$ forms a bounding polygon which contains the region determined by the given periodic path. In the EUP case, it is a bounding line segment. On the other hand in the ESO and EUO cases, we must have $0 < \theta < 90$ since there are no 90 degree angles. However since these reflecting angles are expressed in terms of x, y, 90 and T with integer coefficients, in order to form the bounding polygon we must eliminate T . This can be done as follows. For each reflecting angle of the form $mx + ny + p90 + T$, we can also say that $0 < 90 - mx - ny - p90 - T < 90$ and similarly for reflecting angles of the form $mx + ny + p90 - T$. We then get two sets of angles involving either T or $-T$ and if we add each equation with T to each equation with $-T$ and divide by 2, we end up with a set of linear combinations of x, y and 90 with rational coefficients which lie between 0 and 90 and

hence produce a bounding polygon in these cases. In the EUO case it is a bounding line segment. These are the bounding polygons that we usually use in our calculations.

It is worth noting that the corner bounding polygon equations are included among the angle bounding polygon equations. This is a consequence of the fact that if a poolshot enters a corner A where $\angle A = x$ at an angle θ and bounces n times before it leaves then the angles involved are

$$\begin{aligned}
&\theta,\\
&\theta + x,\\
&\theta + 2x,\\
&...,\\
&180 - \theta - (n-2)x,\\
&180 - \theta - (n-1)x,\\
&180 - \theta - nx
\end{aligned}$$

and then since $0 < \theta < 90$ and $0 < 180 - \theta - nx < 90$, we must have $0 < 180 - nx < 180$ or $0 < nx < 180$, which is one of the corner equations.

8.3 The Program and Proof

Because of the complexity and quantity of these equations, there is a dire need to automate the process of proving the codes work. Thus, we have written a program to crunch the numbers. In these calculations, each G_j is an integer and is exact. On the other hand r, b_1 , and b_2 are in radians and in fact are rational multiples of $\pi/2$, so they too are exact. The f_j involve evaluating sines and cosines, so they are not exact.

All these calculations are done by computer and have a certain degree of accuracy. We need to make sure that when we calculate that $f_j > 0$, this bound is rigorous. To do this we use interval arithmetic which can show that f_j lies exactly within an interval $[u, v]$ with $u > 0$. The interval that we use for $\pi/2$ correct to 7 decimal places is $(1.57079631, 1.57079637)$. This precision can be increased as required. We mainly use the arithmetical operations of

1. addition: $[x_1, x_2] + [y_1, y_2] = [x_1 + y_1, x_2 + y_2]$
2. subtraction: $[x_1, x_2] - [y_1, y_2] = [x_1 - y_2, x_2 - y_1]$
3. multiplication: $[x_1, x_2][y_1, y_2] = [\min(x_1y_1, x_1y_2, x_2y_1, x_2y_2), \max(x_1y_1, x_1y_2, x_2y_1, x_2y_2)]$

Note, we dont use any division operations. In doing the calculations in the prover, we rely on the following two libraries for arbitrary precision arithmetic.

1. GMP: https://gmplib.org/
2. MPFR: http://www.mpfr.org/ [6]

Along with this paper, you should get our program and instructions here using the terminal.

In the instructions of this program, we detail how you can view the proof. You can go square by square to see all the equations that produce the given code region and the lower bound using interval arithmetic of the calculations that show a square satisfies the prover. Because of how massive the proof is, there isnt a good way to present all of the calculations at once, so to trust that the algorithms have been implemented properly, we recommend that you read the code. For those that dont read code and want to feel more comfortable about the proof they could try some random triangles from previous papers. A good one to start with would be [8] and see that it gives the same shape. Others would be [7] and [9].

In Appendix B, we have an enumeration of the 134 code regions that cover the rest of the total region between $z = 75$ and $z = 80$ and which are not part of the two infinite patterns. The endpoints of this region in clockwise order are $(37.5, 37.5)$, $(40, 40)$, $(12.5, 67.5)$, $(7.5, 67.5)$. We used $13,862$ squares at 7 decimal accuracy starting from a single square and had to subdivide any square at most 20 times to be able to prove

that these 134 code regions do cover this region. The enumeration shows the code type, a 2-tuple consisting of the code length and the side sequence length followed by the code sequence.

On this same program, you can find listed the 2439 single codes, 278,131 squares and 21 triples that are used to prove from 105 to 110, the 38,132 single codes, 4,994,538 squares and 310 triples to prove from 110 to 112 and the 118,809 single codes, 27,783,085 squares and 1,115 triples to prove from 112 to 112.3.

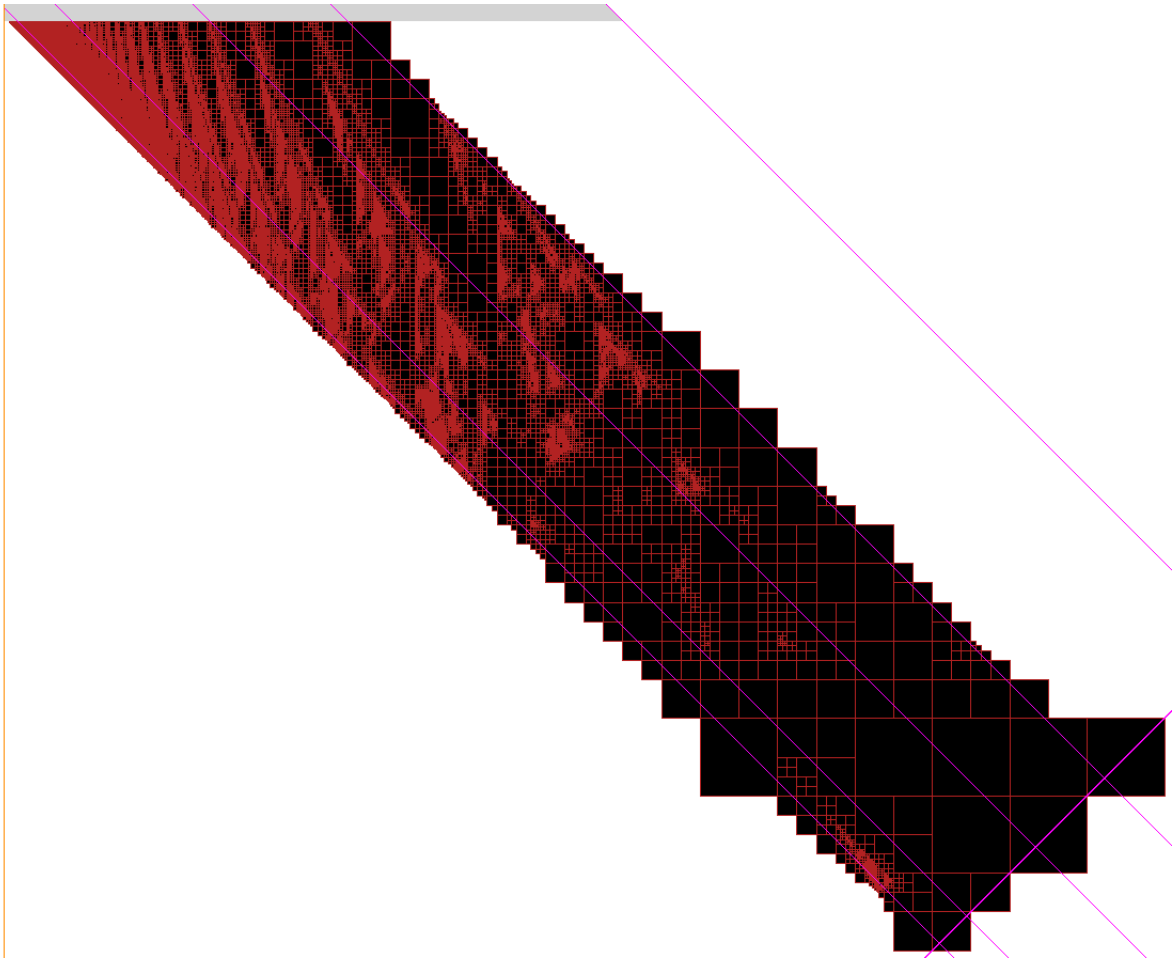


Figure 18: Cover of 100 to 112.3 degrees

From our data, we can see there is a large growth as we go from 112 to 112.3. See Figure 24. This is entirely due to the pile up of codes as we get closer and closer to the point (0, 112.5). All points in the interior between 100 and 112.3 can be surrounded by a finite neighbourhood of codes and patterns, some stable and some unstable.

Acknowledgements

The authors would like to thank the NSERC USRA program, Gerald Cliff, Brendan Pass, Eric Woolgar, and Terry Gannon, all of whom provided financial support to this project.

References

[1] Fred Holt, "Periodic Reflecting Paths in Right Triangles", *Geometriae Dedicata* 46, no. 1 (April 1993): 73–90.

[2] Howard Masur, "Closed Trajectories for Quadratic Differentials with an Application to Billiards", *Duke Mathematical Journal* 53, no. 2 (1986): 307–314.

[3] Richard Evan Schwartz, "Obtuse Triangular Billiards II: One Hundred Degrees Worth of Periodic Trajectories", *Experimental Mathematics* 18, no. 2 (2009): 137–171.

[4] W. Patrick Hooper, Richard Evan Schwartz, "Billiards in Nearly Isosceles Triangles", Journal of Modern Dynamics 3, no. 2 (2009): 159–231.

[5] Torbjrn Granlund et al., GNU Multiple Precision Arithmetic Library, https://gmplib.org.

[6] Laurent Fousse, Guillaume Hanrot, Vincent Lefvre, Patrick Plissier, Paul Zimmermann, "MPFR: A Multiple-Precision Binary Floating-Point Library With Correct Rounding", *ACM Transactions on Mathematical Software* 33, no. 2 (June 2007): Article 13.

[7] Ya. B. Vorobets, G. A. Galperin, and A. M. Stepin, Periodic billiard trajectories in polygons generating mechanisms Russ. Math. Surv. 47, no. 5 (1992): 5-80.

[8] W. Patrick Hooper, Richard Evan Schwartz, Introduction to McBilliards Geometriae Dedicata 125, no. 1 (Sept. 2004): 39-46.

[9] Lorenz Halbeisen and Norbert Hungerbuhler, On periodic billiard trajectories in obtuse triangles SIAM review 42, no. 4 (2000): 657-670.

A Example of Calculating Coordinates of a Code Tower

Here we will calculate the vertices in the code tower of the code EUO code sequence along the axis $x = y$.

$$\begin{array}{cccccc} X & & Z & & X & \\ 1 & 1 & 2 & 3 & 3 & 2 \\ & Y & & Y & & Z \end{array}$$

- $x_1 = \sin y$, $y_1 = 0$ which are the coordinates of $L_{(1,0)}$
- $x_2 = 0$, $y_2 = 0$ which are the coordinates of $L_{(2,0)}$
- $x_3 = \sin z \cos X$, $y_3 = \sin z \sin X$ which are the coordinates of $L_{(3,0)}$
- $x_4 = \sin z \cos X - \sin x \cos(Y - X)$, $y_4 = \sin z \sin X + \sin x \sin(Y - X)$ which are the coordinates of $L_{(4,0)}$
- $x_5 = \sin z \cos X - \sin x \cos(Y - X) + \sin x \cos(2Z + X - Y)$,
  $y_5 = \sin z \sin X + \sin x \sin(Y - X) + \sin x \sin(2Z + X - Y)$ which are the coordinates of $L_{(5,0)}$
- $x_6 = \sin z \cos X - \sin x \cos(Y - X) + \sin x \cos(2Z + X - Y) - \sin z \cos(3Y - 2Z + Y - X)$,
  $y_6 = \sin z \sin X + \sin x \sin(Y - X) + \sin x \sin(2Z + X - Y) + \sin z \sin(3Y - 2Z + Y - X)$ which are the coordinates of $L_{(6,0)}$
- $x_7 = \sin z \cos X - \sin x \cos(Y - X) + \sin x \cos(2Z + X - Y) - \sin z \cos(3Y - 2Z + Y - X) + \sin y \cos(3X - 3Y + 2Z - Y + X)$,
  $y_7 = \sin z \sin X + \sin x \sin(Y - X) + \sin x \sin(2Z + X - Y) + \sin z \sin(3Y - 2Z + Y - X) + \sin y \sin(3X - 3Y + 2Z - Y + X)$ which are the coordinates of $L_{(7,0)}$
- $x_8 = \sin z \cos X - \sin x \cos(Y - X) + \sin x \cos(2Z + X - Y) - \sin z \cos(3Y - 2Z + Y - X) + \sin y \cos(3X - 3Y + 2Z - Y + X) - \sin y \cos(2Z - 3X + 3Y - 2Z + Y - X)$,
  $y_8 = \sin z \sin X + \sin x \sin(Y - X) + \sin x \sin(2Z + X - Y) + \sin z \sin(3Y - 2Z + Y - X) + \sin y \sin(3X - 3Y + 2Z - Y + X) + \sin y \sin(2Z - 3X + 3Y - 2Z + Y - X)$ which are the coordinates of $L_{(8,0)}$

We then use standard trig identities together with the conditions that $z = 180 - x - y$ and $y = x$ from the code pattern.

The shooting vector is (c, d) where (c, d) is the vector from $L_{(2,0)}$ to $L_{(8,0)}$,

$$\begin{aligned} c = \sin(y)\cos(180) + \sin(z)\cos(x) + \sin(x)\cos(x - y + 180) + \sin(x)\cos(-x - 3y + 4(180)) \\ + \sin(z)\cos(-x - 6y + 5(180)) + \sin(y)\cos(2x - 6y + 6(180)) \\ d = \sin(y)\sin(180) + \sin(z)\sin(x) + \sin(x)\sin(x - y + 180) + \sin(x)\sin(-x - 3y + 4(180)) \\ + \sin(z)\sin(-x - 6y + 5(180)) + \sin(y)\sin(2x - 6y + 6(180)) \end{aligned}$$

The convention is to use trig identities to simplify to sums of sines and cosines, and then we multiply all coordinates by 2 so that all coefficients are integers.

The shooting vector (c, d) then further becomes

$$\begin{aligned} c = -2sin(y) - \sin(3y) + \sin(5y) - \sin(2x - 7y) + \sin(2x - 5y) - \sin(2x - y) + \sin(2x + y) \\ + \sin(2x + 3y) - \sin(2x + 7y) \end{aligned}$$

a sum of sines, and

$$\begin{aligned} d = -\cos(3y) + \cos(5y) + \cos(2x - 7y) - \cos(2x - 5y) + \cos(2x - y) - \cos(2x + y) \\ + \cos(2x + 3y) - \cos(2x + 7y) \end{aligned}$$

a sum of cosines.

To calculate the coordinates of the key points in the tower which are not centers of fans if any, we illustrate by the same example. Suppose we look at the coordinates of $L_{(4,1)}$. It is found by starting with the coordinates of $L_{(4,0)}$ and since the corresponding code is 2 adding one more reflection of the given triangle and proceeding as before.

$$x_4 = \sin z \cos X - \sin x \cos(Y - X),$$
$$y_4 = \sin z \sin X + \sin x \sin(Y - X)$$

becomes

$$x_{(4,1)} = \sin z \cos X - \sin x \cos(Y - X) + \sin y \cos(Z - Y + X),$$
$$y_{(4,1)} = \sin z \sin X + \sin x \sin(Y - X) + \sin y \sin(Z - Y + X)$$

and we then simplify to sums of sines or cosines.

To find the coordinates of $L_{(6,2)}$, we would start with the coordinates of $L_{(6,0)}$ and since the corresponding code is 3 adding two more reflections of the given triangle which corresponds to using the angle $2X - 3Y + 2Z - Y + X$ and we get

$$x_{(6,2)} = x_6 + \sin z \cos(2X - 3Y + 2Z - Y + X) = x_6 + \sin z \cos(x - 6y)$$
$$y_{(6,2)} = y_6 + \sin z \sin(2X - 3Y + 2Z - Y + X) = x_6 + \sin z \sin(x - 6y)$$

and we then simplify to sums of sines or cosines.

It is now a simple matter to calculate a vector (a, b) from any key blue point to any key black point. As an example, the blue-black vector from $L_{(6,0)}$ to $L_{(5,0)}$ is given by

$$a = \sin z \cos(3Y - 2Z + Y - X) = \sin z \cos(6y + x) = \sin(7y + 2x) - \sin 5x = \sin 9x - \sin 5x$$
$$b = -\sin z \sin(3Y - 2Z + Y - X) = -\sin z \sin(6y + x) = \cos(7y + 2x) - \cos 5y = \cos 9x - \cos 5x$$

since $y = x$ for this code.

B The Singles for the 105 theorem

1. OSO (3, 7) 1 3 3
2. OSO (5, 11) 1 1 2 2 5
3. OSO (5, 15) 1 1 4 2 7
4. OSO (5, 15) 1 3 2 6 3
5. OSO (5, 17) 1 1 4 2 9
6. OSO (5, 21) 1 1 6 2 11
7. OSO (5, 23) 1 1 6 2 13
8. OSO (7, 15) 1 1 3 1 2 1 6
9. OSO (7, 17) 1 1 3 1 2 1 8
10. OSO (7, 19) 1 1 2 2 6 2 5
11. OSO (7, 21) 1 1 2 2 8 2 5
12. OSO (7, 23) 1 1 4 2 6 2 7
13. OSO (7, 29) 1 1 6 2 8 2 9
14. OSO (7, 17) 1 2 1 2 1 3 7
15. ESO (8, 18) 1 1 2 2 3 1 3 5
16. ESO (8, 22) 1 1 4 2 3 1 3 7
17. OSO (9, 25) 1 1 2 2 6 2 4 2 5
18. ESP (10, 20) 1 1 3 1 2 1 3 1 1 6
19. ESO (10, 24) 1 1 3 1 2 1 5 1 1 8
20. ESP (10, 28) 1 1 5 1 2 1 5 1 1 10
21. ESO (10, 32) 1 1 5 1 2 1 7 1 1 12
22. ESP (10, 36) 1 1 7 1 2 1 7 1 1 14

23. ESO (10, 40) 1 1 7 1 2 1 9 1 1 16
24. ESP (10, 44) 1 1 9 1 2 1 9 1 1 18
25. ESO (10, 48) 1 1 9 1 2 1 11 1 1 20
26. ESO (10, 26) 1 1 2 2 7 1 1 4 2 5
27. ESO (10, 38) 1 1 4 2 11 1 1 6 2 9
28. OSO (11, 29) 1 1 2 1 1 7 2 4 1 1 8
29. OSO (11, 25) 1 1 2 1 1 6 1 1 2 2 7
30. ESO (12, 38) 1 1 2 2 8 2 3 1 3 8 2 5
31. ESO (14, 40) 1 1 3 1 2 1 7 2 4 1 1 7 2 7
32. ESO (14, 44) 1 1 3 1 2 1 7 2 6 1 1 9 2 7
33. ESO (14, 52) 1 1 3 1 2 1 11 2 6 1 1 11 2 9
34. ESO (14, 36) 1 1 2 2 7 1 2 1 3 1 1 7 2 5
35. ESP (14, 36) 1 2 1 5 3 1 3 2 4 2 3 1 3 5
36. ESP (16, 44) 1 1 2 1 1 7 3 1 3 2 6 2 3 1 3 7
37. ESP (16, 52) 1 1 4 1 1 9 3 1 3 2 8 2 3 1 3 9
38. ESO (18, 42) 1 1 2 1 1 6 1 1 2 2 7 1 2 1 2 1 3 7
39. ESP (18, 44) 1 1 2 2 5 1 2 1 5 2 2 1 1 5 2 4 2 5
40. ESP (18, 52) 1 1 4 2 5 1 2 1 5 2 4 1 1 7 2 4 2 7
41. OSO (19, 57) 1 1 2 1 1 6 1 2 1 3 1 1 7 2 8 2 8 2 7
42. ESP (20, 48) 1 1 3 1 2 1 6 1 2 1 3 1 1 8 1 1 4 1 1 8
43. ESP (20, 56) 1 1 3 1 2 1 10 1 2 1 3 1 1 10 1 1 4 1 1 10
44. ESP (20, 64) 1 1 5 1 2 1 8 1 2 1 5 1 1 12 1 1 6 1 1 12
45. ESP (20, 88) 1 1 9 1 2 1 8 1 2 1 9 1 1 18 1 1 10 1 1 18
46. ESO (20, 74) 1 1 2 1 1 7 1 1 13 2 7 1 2 1 7 1 1 13 2 9
47. ESP (20, 52) 1 1 2 1 1 7 2 2 1 1 5 2 6 2 5 1 1 2 2 7
48. ESP (20, 60) 1 1 2 1 1 7 2 4 1 1 7 2 6 2 7 1 1 4 2 7
49. ESP (20, 68) 1 1 4 1 1 9 2 4 1 1 7 2 8 2 7 1 1 4 2 9
50. ESP (20, 92) 1 1 6 1 1 13 2 6 1 1 11 2 10 2 11 1 1 6 2 13
51. ESP (20, 60) 1 1 2 2 9 1 1 4 2 8 2 4 1 1 9 2 2 1 1 6
52. ESP (20, 92) 1 1 6 2 13 1 1 8 2 10 2 8 1 1 13 2 6 1 1 12
53. ESP (20, 60) 1 1 4 2 6 2 4 1 1 7 2 4 1 1 8 1 1 4 2 7
54. OSO (21, 49) 1 1 1 1 2 1 7 2 5 1 1 2 1 1 7 2 5 1 2 1 4
55. ESO (22, 54) 1 1 1 1 3 7 1 1 3 1 2 1 5 2 6 2 2 1 1 5 2 5
56. ESO (22, 82) 1 1 2 1 1 7 1 1 12 1 1 4 1 1 11 2 8 1 1 13 2 9
57. ESO (22, 58) 1 1 2 2 7 1 2 1 3 2 7 1 1 3 1 2 1 5 2 6 2 5
58. ESO (22, 64) 1 1 2 2 7 1 2 1 3 2 6 2 4 1 1 7 2 4 2 6 2 5
59. ESO (22, 60) 1 1 4 2 7 1 2 1 4 1 2 1 5 1 1 9 2 3 1 2 1 8
60. ESO (22, 74) 1 1 2 2 8 2 4 2 6 2 7 1 1 4 2 6 2 6 2 6 2 5
61. OSO (23, 55) 1 1 2 1 1 5 1 1 8 1 2 1 2 1 2 1 8 1 1 4 1 1 8
62. ESP (24, 64) 1 1 1 1 2 1 9 3 1 2 1 3 9 1 2 1 1 1 1 5 2 8 2 5
63. ESO (24, 60) 1 1 3 1 2 1 5 2 3 1 3 6 2 3 1 3 7 1 1 3 1 2 1 6
64. ESO (24, 88) 1 1 3 1 2 1 11 2 9 1 2 1 5 1 1 12 1 1 6 1 1 13 2 9
65. ESO (24, 68) 1 1 3 1 2 1 8 1 1 4 1 1 9 2 4 1 1 9 2 3 1 2 1 8
66. ESP (24, 64) 1 1 2 1 1 5 1 1 10 1 1 6 1 1 10 1 1 5 1 1 2 1 1 8
67. ESP (24, 76) 1 1 4 2 6 2 3 1 3 8 3 1 3 2 6 2 4 1 1 7 2 4 2 7
68. ESP (26, 64) 1 1 1 1 2 1 7 2 4 2 7 1 2 1 1 1 1 5 2 5 1 2 1 5 2 5
69. ESP (26, 104) 1 1 5 1 2 1 9 2 12 2 9 1 2 1 5 1 1 13 2 7 1 2 1 7 2 13
70. ESO (26, 72) 1 1 2 1 1 5 1 1 8 1 2 1 3 2 9 1 1 4 2 9 1 1 4 1 1 8
71. ESP (26, 64) 1 1 2 1 1 6 1 1 2 1 1 7 3 1 3 2 5 1 2 1 5 2 3 1 3 7
72. ESP (26, 124) 1 1 6 2 12 2 7 1 2 1 7 2 12 2 6 1 1 11 2 8 2 12 2 8 2 11
73. ESP (28, 80) 1 1 3 1 2 1 7 2 4 1 1 8 1 1 4 2 7 1 2 1 3 1 1 7 2 6 2 7
74. ESP (28, 112) 1 1 3 1 2 1 11 2 8 1 1 14 1 1 8 2 11 1 2 1 3 1 1 9 2 12 2 9
75. ESO (28, 118) 1 1 3 1 2 1 11 2 8 2 10 2 11 1 1 6 2 12 2 8 2 11 1 1 4 1 1 10
76. ESP (28, 76) 1 1 3 1 2 1 10 1 1 4 1 1 10 1 2 1 3 1 1 8 1 2 1 6 1 2 1 8

77. ESP (28, 80) 1 1 2 1 1 5 1 1 9 2 4 2 9 1 1 5 1 1 2 1 1 9 3 1 2 1 3 9
78. ESO (28, 74) 1 1 2 1 1 5 2 6 2 5 1 1 2 2 7 1 2 1 3 2 6 2 5 1 1 2 2 7
79. ESO (28, 68) 1 1 2 2 6 2 5 1 1 2 2 7 1 1 3 1 2 1 6 1 1 3 1 2 1 5 2 5
80. ESP (28, 88) 1 1 4 2 10 2 4 1 1 7 2 9 1 1 5 1 2 1 4 1 2 1 5 1 1 9 2 7
81. ESP (28, 92) 1 1 4 2 6 2 6 2 6 2 4 1 1 7 2 6 2 4 1 1 8 1 1 4 2 6 2 7
82. ESO (30, 76) 1 1 2 2 5 1 2 1 4 1 2 1 5 3 1 2 1 3 6 3 1 3 2 4 2 4 2 4 2 5
83. OSO (31, 87) 1 1 3 1 2 1 6 1 2 1 3 1 1 7 2 6 2 4 1 1 7 2 5 1 2 1 5 2 6 2 7
84. ESO (32, 84) 1 1 2 1 1 5 1 1 8 1 1 2 1 1 5 2 6 2 7 1 1 4 2 7 1 1 2 1 1 5 2 7
85. ESP (32, 76) 1 1 2 1 1 5 1 1 8 1 2 1 2 1 2 1 8 1 1 5 1 1 2 1 1 8 1 1 4 1 1 8
86. ESP (32, 76) 1 1 2 1 1 6 1 2 1 3 1 1 8 1 1 4 1 1 8 1 1 4 1 1 8 1 1 3 1 2 1 6
87. ESP (32, 80) 1 1 2 2 5 1 2 1 5 3 1 2 1 3 5 1 2 1 5 2 2 1 1 5 2 4 2 4 2 4 2 5
88. ESO (34, 80) 1 1 1 1 3 7 1 1 3 1 2 1 5 2 5 1 2 1 3 2 7 1 1 3 1 2 1 5 2 5 1 2 1 4
89. ESP (34, 96) 1 1 3 1 2 1 5 2 6 2 4 2 6 2 5 1 2 1 3 1 1 7 2 4 2 5 1 2 1 5 2 4 2 7
90. ESP (36, 84) 1 1 2 1 1 5 1 1 8 1 1 2 1 1 6 1 1 2 1 1 8 1 1 5 1 1 2 1 1 8 1 1 4 1 1 8
91. ESP (36, 108) 1 1 2 1 1 7 1 1 12 1 1 4 1 1 10 1 1 4 1 1 12 1 1 7 1 1 2 1 1 10 1 1 4 1 1 10
92. ESP (36, 132) 1 1 4 1 1 9 1 1 14 1 1 6 1 1 12 1 1 6 1 1 14 1 1 9 1 1 4 1 1 12 1 1 6 1 1 12
93. ESP (36, 84) 1 1 2 1 1 6 1 1 2 1 1 6 1 1 2 1 1 7 3 1 3 2 5 1 2 1 4 1 2 1 5 2 3 1 3 7
94. ESP (36, 84) 1 1 2 1 1 6 1 1 2 2 7 1 2 1 2 1 3 7 1 1 2 1 1 7 3 1 2 1 2 1 7 2 2 1 1 6
95. ESO (36, 102) 1 1 2 1 1 6 1 2 1 5 2 7 1 1 3 1 2 1 6 1 2 1 5 2 6 2 6 2 7 1 1 4 2 6 2 7
96. ESP (36, 112) 1 1 2 2 9 1 1 4 1 1 9 3 1 3 2 8 2 4 2 8 2 3 1 3 9 1 1 4 1 1 9 2 2 1 1 6
97. ESP (38, 100) 1 1 2 1 1 6 1 1 2 1 1 7 2 7 1 1 3 1 2 1 7 2 5 1 2 1 5 2 7 1 2 1 3 1 1 7 2 7
98. ESO (40, 96) 1 1 1 1 3 7 1 1 3 1 2 1 5 2 5 1 1 2 2 7 1 1 3 1 2 1 5 2 6 2 2 1 1 5 2 5 1 2 1 4
99. ESO (40, 128) 1 1 3 1 2 1 10 1 1 5 1 2 1 7 2 13 1 1 7 1 2 1 6 1 2 1 7 2 13 1 1 7 1 2 1 6 1 2 1 8
100. ESO (40, 120) 1 1 3 1 2 1 8 1 1 4 2 8 2 5 1 2 1 5 1 1 9 2 5 1 2 1 5 2 8 2 4 1 1 9 2 3 1 2 1 8
101. ESP (40, 148) 1 1 2 1 1 7 1 1 13 2 9 1 2 1 5 1 1 13 2 8 2 13 1 1 5 1 2 1 9 2 13 1 1 7 1 1 2 1 1 10
102. ESP (40, 120) 1 1 2 1 1 7 2 4 1 1 7 2 6 2 6 2 6 2 7 1 1 4 2 7 1 1 2 1 1 7 2 4 1 1 8 1 1 4 2 7
103. ESP (40, 100) 1 1 2 1 1 6 1 1 3 1 2 1 5 2 6 2 5 1 2 1 3 1 1 6 1 1 2 1 1 7 2 4 1 1 8 1 1 4 2 7
104. ESO (42, 116) 1 1 2 1 1 5 1 1 9 2 4 2 9 1 1 4 2 9 1 1 4 1 1 8 1 2 1 3 1 1 8 1 1 2 1 1 6 1 1 2 2 9
105. ESO (44, 118) 1 1 2 1 1 7 2 4 1 1 8 1 1 2 1 1 7 2 4 1 1 8 1 2 1 3 2 9 1 1 5 1 2 1 4 1 2 1 7 2 4 1 1 8
106. ESP (48, 140) 1 1 1 1 3 9 1 1 2 1 1 5 1 1 9 2 4 2 9 1 1 4 1 1 9 2 4 2 9 1 1 5 1 1 2 1 1 9 3 1 1 1 1 5 2 8 2 5
107. ESP (48, 136) 1 1 2 1 1 6 1 1 2 1 1 7 2 2 1 1 5 2 6 2 4 2 7 1 1 4 2 6 2 4 2 6 2 4 1 1 7 2 4 2 6 2 5 1 1 2 2 7
108. ESP (50, 156) 1 1 1 1 3 9 1 1 4 2 8 2 5 1 2 1 5 2 8 2 5 1 2 1 5 2 8 2 4 1 1 9 3 1 1 1 1 5 2 8 2 5 1 2 1 5 2 8 2 5
109. ESO (50, 156) 1 1 2 1 1 7 2 6 2 6 2 8 2 4 1 1 9 2 3 1 2 1 8 1 1 3 1 2 1 8 1 1 4 2 8 2 6 2 6 2 7 1 2 1 3 1 1 7 2 7
110. ESP (56, 132) 1 1 1 1 2 1 7 2 4 1 1 8 1 1 4 2 7 1 2 1 1 1 1 4 1 2 1 5 2 7 1 1 3 1 2 1 6 1 1 2 1 1 6 1 2 1 3 1 1 7 2 5 1 2 1 4
111. ESP (56, 160) 1 1 2 1 1 5 1 1 9 2 4 2 9 1 1 4 1 1 9 2 4 2 9 1 1 5 1 1 2 1 1 9 3 1 2 1 3 9 1 1 2 1 1 6 1 1 2 1 1 9 3 1 2 1 3 9
112. ESO (56, 216) 1 1 4 1 1 9 2 11 1 1 5 1 2 1 8 1 2 1 5 1 1 12 1 1 6 1 1 13 2 6 2 13 1 1 6 2 13 1 1 6 1 1 12 1 1 4 1 1 9 2 12 2 6 1 1 12
113. ESP (56, 216) 1 1 2 1 1 8 1 1 2 1 1 9 2 12 2 9 1 2 1 5 1 1 13 2 7 1 2 1 7 2 13 1 1 6 1 1 13 2 7 1 2 1 7 2 13 1 1 5 1 2 1 9 2 12 2 9
114. ESO (56, 142) 1 1 2 1 1 6 1 2 1 3 1 1 7 2 5 1 2 1 4 1 2 1 6 1 1 2 1 1 7 2 5 1 1 2 2 8 2 5 1 1 2 2 9 1 1 5 1 2 1 4 1 2 1 5 2 7
115. ESO (56, 170) 1 1 2 1 1 6 1 2 1 5 2 8 2 4 1 1 8 1 2 1 3 1 1 9 2 2 1 1 5 1 1 9 2 4 1 1 9 2 4 2 8 2 6 2 6 2 7 1 2 1 3 1 1 7 2 7
116. ESP (56, 172) 1 1 2 2 9 1 1 4 1 1 9 2 2 1 1 6 1 1 2 2 9 1 1 4 1 1 9 3 1 3 2 8 2 4 2 8 2 4 2 8 2 3 1 3 9 1 1 4 1 1 9 2 2 1 1 6
117. ESO (58, 158) 1 1 2 1 1 6 1 2 1 5 2 7 1 2 1 3 1 1 7 2 7 1 1 2 1 1 6 1 2 1 5 2 8 2 4 1 1 9 2 2 1 1 5 1 1 9 2 4 1 1 8 1 2 1 3 1 1 8
118. ESO (60, 162) 1 1 2 1 1 5 1 1 9 2 4 2 9 1 1 4 2 9 1 1 4 1 1 9 2 3 1 2 1 8 1 2 1 2 1 3 9 1 1 2 1 1 6 1 1 2 1 1 8 1 1 2 1 1 6 1 1 2 2 9
119. ESP (60, 144) 1 1 2 1 1 5 1 1 8 1 2 1 2 1 2 1 8 1 1 4 1 1 8 1 2 1 2 1 2 1 8 1 1 5 1 1 2 1 1 8 1 1 4 1 1 8 1

2 1 2 1 2 1 8 1 1 4 1 1 8

120. ESP (60, 180) 1 1 2 1 1 7 2 4 1 1 7 2 6 2 6 2 6 2 6 2 6 2 7 1 1 4 2 7 1 1 2 1 1 7 2 4 1 1 8 1 1 4 2 7 1 1 2 1 1 7 2 4 1 1 8 1 1 4 2 7
121. ESO (60, 186) 1 1 2 1 1 7 2 6 2 6 2 8 2 4 1 1 8 1 2 1 3 1 1 8 1 2 1 3 2 9 1 1 4 2 8 2 6 2 6 2 7 1 1 2 1 1 7 2 6 2 7 1 2 1 3 1 1 7 2 7
122. ESP (64, 156) 1 1 2 1 1 6 1 1 2 1 1 8 1 1 3 1 2 1 7 2 6 2 7 1 2 1 3 1 1 8 1 1 2 1 1 6 1 1 2 1 1 9 2 2 1 1 6 1 1 2 1 1 8 1 1 2 1 1 6 1 1 2 2 9
123. ESP (66, 200) 1 1 1 1 3 9 1 1 2 1 1 5 1 1 9 2 4 2 9 1 1 4 1 1 9 2 4 2 9 1 1 4 1 1 9 2 4 2 9 1 1 5 1 1 2 1 1 9 3 1 1 1 1 5 2 8 2 5 1 2 1 5 2 8 2 5
124. ESP (66, 208) 1 1 1 1 3 9 1 1 4 2 8 2 5 1 2 1 5 2 8 2 5 1 2 1 5 2 8 2 5 1 2 1 5 2 8 2 4 1 1 9 3 1 1 1 1 5 2 8 2 5 1 2 1 5 2 8 2 5 1 2 1 5 2 8 2 5
125. ESO (66, 230) 1 1 2 1 1 5 1 1 9 2 4 1 1 9 2 4 2 8 2 6 2 6 2 8 2 4 1 1 9 2 4 2 8 2 6 2 6 2 8 2 4 2 9 1 1 4 1 1 9 2 4 1 1 9 2 4 2 9 1 1 5 1 1 2 2 9
126. ESP (68, 188) 1 1 2 1 1 6 1 1 2 1 1 8 1 2 1 3 2 9 1 1 4 2 8 2 5 1 2 1 6 1 1 2 1 1 8 1 1 4 2 8 2 4 1 1 8 1 1 2 1 1 6 1 2 1 5 2 8 2 4 1 1 9 2 3 1 2 1 8
127. ESP (72, 216) 1 1 2 1 1 7 2 4 1 1 8 1 1 2 1 1 7 2 4 2 8 2 4 2 8 2 4 2 7 1 1 2 1 1 8 1 1 4 2 7 1 1 2 1 1 7 2 5 1 2 1 5 2 8 3 1 3 2 8 2 3 1 3 8 2 5 1 2 1 5 2 7
128. ESO (74, 220) 1 1 2 1 1 6 1 2 1 5 2 8 2 4 2 9 1 1 4 2 8 2 6 2 6 2 7 1 1 2 1 1 7 2 7 1 1 3 1 2 1 7 2 6 2 7 1 1 3 1 2 1 7 2 5 1 2 1 5 1 1 9 2 4 1 1 8 1 2 1 3 1 1 8
129. ESP (80, 324) 1 1 2 1 1 7 2 13 1 1 8 2 10 2 9 1 2 1 5 1 1 12 1 1 6 2 13 1 1 6 1 1 12 1 1 6 2 12 2 8 2 12 2 6 1 1 12 1 1 6 1 1 13 2 6 1 1 12 1 1 5 1 2 1 9 2 10 2 8 1 1 13 2 7 1 1 2 1 1 10
130. ESP (84, 260) 1 1 1 1 3 9 1 1 2 1 1 5 1 1 9 2 4 2 9 1 1 4 1 1 9 2 4 2 9 1 1 4 1 1 9 2 4 2 9 1 1 4 1 1 9 2 4 2 9 1 1 5 1 1 2 1 1 9 3 1 1 1 1 5 2 8 2 5 1 2 1 5 2 8 2 5 1 2 1 5 2 8 2 5
131. ESP (92, 252) 1 1 1 1 3 9 1 1 5 1 1 2 1 1 8 1 1 4 1 1 8 1 2 1 2 1 2 1 8 1 1 4 1 1 8 1 1 2 1 1 5 1 1 9 3 1 1 1 1 5 2 6 2 7 1 2 1 2 1 2 1 8 1 1 4 2 8 2 4 2 8 2 4 2 8 2 4 1 1 8 1 2 1 2 1 2 1 7 2 6 2 5
132. ESP (100, 300) 1 1 2 1 1 7 2 4 1 1 7 2 6 2 6 2 6 2 6 2 6 2 6 2 6 2 6 2 6 2 7 1 1 4 2 7 1 1 2 1 1 7 2 4 1 1 8 1 1 4 2 7 1 1 2 1 1 7 2 4 1 1 8 1 1 4 2 7 1 1 2 1 1 7 2 4 1 1 8 1 1 4 2 7 1 1 2 1 1 7 2 4 1 1 8 1 1 4 2 7
133. ESP (102, 320) 1 1 1 1 3 9 1 1 2 1 1 5 1 1 9 2 4 2 9 1 1 4 1 1 9 2 4 2 9 1 1 4 1 1 9 2 4 2 9 1 1 4 1 1 9 2 4 2 9 1 1 4 1 1 9 2 4 2 9 1 1 5 1 1 2 1 1 9 3 1 1 1 1 5 2 8 2 5 1 2 1 5 2 8 2 5 1 2 1 5 2 8 2 5 1 2 1 5 2 8 2 5
134. ESO (116, 326) 1 1 2 1 1 6 1 2 1 5 2 7 1 2 1 4 1 2 1 6 1 2 1 5 2 7 1 1 2 1 1 8 1 1 4 2 8 2 5 1 2 1 5 2 8 2 5 1 2 1 5 2 9 1 1 5 1 2 1 5 2 8 2 5 1 2 1 5 2 8 2 4 1 1 8 1 1 2 1 1 7 2 5 1 2 1 6 1 2 1 4 1 2 1 7 2 5 1 2 1 5 1 1 9 2 4 1 1 8 1 2 1 3 1 1 8