

HYPERBOLIC SPLINE CURVES USING A WEIGHTED AVERAGE

Yanhua Li

Advisor: Dr. Douglas Dunham

Department of Computer Science
University of Minnesota Duluth
Duluth, Minnesota 55812

HYPERBOLIC SPLINE CURVES USING A WEIGHTED AVERAGE

A Thesis

submitted to the Faculty of the Graduate School
of the University of Minnesota

by

Yanhua Li

In partial fulfillment of the
requirements for the degree of
Master of Science
in Computer Science

August 2004

UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of a master's thesis by

Yanhua Li

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Dr. Douglas Dunham

Name of Faculty Adviser

Signature of Faculty Adviser

Date

GRADUATE SCHOOL

Acknowledgements

First of all, I would like to thank my advisor Dr. Douglas Dunham. It is he who introduced me to this interesting project. He advised me and provided me support in every step of my research. I learned a lot of valuable knowledge from him. I also want to thank Dr. Donald Crouch and Dr. Guihua Fei for kindly agreeing to serve as my committee members and being so nice to students. They also made many helpful comments and corrections to my paper. Finally, I would like to thank my parents for their endless love and support in my life.

Abstract

Currently there is no theory of spline curves in hyperbolic space. However there are well developed theories of spline curves in Euclidean space. There are also methods for specifying splines on spheres, among which the recently described weighted averages method of Buss and Fillmore seems to overcome the shortcomings of the previous methods.

In this project, we designed an algorithm that can draw natural analogues of Euclidean and spherical spline curves in the hyperboloid using a weighted average method similar to that of Buss and Fillmore. We then implemented this algorithm in a graphics program. Since the weighted average splines have been defined in Euclidean and spherical space, this research would fill a gap by defining such splines in hyperbolic space, the third of the “classical” geometries.

Contents

1	Introduction	1
2	Hyperbolic Geometry	2
2.1	Basic Definitions	2
2.2	Models	3
2.2.1	The Weierstrass Model	4
2.2.2	The Poincaré Disk Model	6
2.2.3	Isomorphism between the Weierstrass and the Poincaré Disk Model	7
3	Parametric Curves	9
3.1	Representations of a Curve in 3-Dimensional Euclidean Space	9
3.2	Parametric Cubic Curves	10
3.3	Bézier Curves	11
4	The Weighted Average Algorithm for Hyperbolic Splines	13
4.1	Hyperbolic Isometric Transformations	13
4.2	Algorithm for Hyperbolic Weighted Averages	19
4.3	Transformation between the Visual Plane and the Hyperboloid	22
4.4	C Program to Calculate Hyperbolic Spline Data	23
5	Results and Analyses	24
5.1	Splines and Their Tangents at Ends	24
5.2	Hyperbolic and Euclidean Splines	40
5.3	Observation	51
5.4	Invariance	51
5.5	An Application with the Hyperbolic Fish Pattern	58
6	Possible Future Work in this Area	62

7 Conclusion
References
Appendix

List of Figures

1	A “line” in the Weierstrass Model[5]	4
2	The Weierstrass Model[5]	5
3	The Poincaré Disk Model[3]	6
4	Isomorphism between the Weierstrass (H^2) and the Poincaré Disk Model (D)[3]	7
5	Two Bézier Curves[2]	11
6	Point p on the Hyperboloid	16
7	Spline 1	25
8	Hyperbolic Spline (red) and Hyperbolic Polyline (blue)	26
9	Euclidean Spline (green) and Euclidean Polyline (magenta)	27
10	Spline 2	28
11	Hyperbolic Spline (red) and Hyperbolic Polyline (blue)	29
12	Euclidean Spline (green) and Euclidean Polyline (magenta)	30
13	Spline 3	31
14	Hyperbolic Spline (red) and Hyperbolic Polyline (blue)	32
15	Euclidean Spline (green) and Euclidean Polyline (magenta)	33
16	Spline 4	34
17	Hyperbolic Spline (red) and Hyperbolic Polyline (blue)	35
18	Euclidean Spline (green) and Euclidean Polyline (magenta)	36
19	Spline 5	37
20	Hyperbolic Spline (red) and Hyperbolic Polyline (blue)	38
21	Euclidean Spline (green) and Euclidean Polyline (magenta)	39
22	Spline 6	41
23	Hyperbolic Spline (red) and Euclidean Spline (green)	42
24	Spline 7	43
25	Hyperbolic Spline (red) and Euclidean Spline (green)	44
26	Spline 8	45

27	Hyperbolic Spline (red) and Euclidean Spline (green)	46
28	Spline 9	47
29	Hyperbolic Spline (red) and Euclidean Spline (green)	48
30	Spline 10	49
31	Hyperbolic Spline (red) and Euclidean Spline (green)	50
32	Spline 11	52
33	Spline 12	53
34	Spline 13	54
35	Spline 14	55
36	Spline 15 using Method 1	56
37	Spline 16 using Method 2	57
38	Original Fish Pattern	59
39	Original Fish Pattern with Four Control Points	60
40	New Fish Pattern with Splines	61

1 Introduction

Currently there is no theory of spline curves in hyperbolic space. Over the years, theories of spline curves in Euclidean space have been well developed. There is also some work on spline curves on spheres (mostly on the 2- and 3- sphere).

Until recently, the splines considered for use on spheres had shortcomings. In 2002, mathematicians Samuel Buss and Jay Fillmore used the weighted average method for spherical splines [1]. Their method uses the weighted average of spherical distances to control points. It is a natural analogue of that for Euclidean splines and it overcomes the shortcomings of the previous methods.

In 2002, a UMD Computer Science graduate student, Amit Lath, investigated some techniques for generating spline curves in the hyperbolic plane [3]. These methods were similar to the methods used before [1] to produce spherical splines. One such method is to generate a Euclidean spline (which doesn't lie on the sphere) and then renormalize it to lie on the sphere. Such methods work acceptably well for splines that are small compared to the diameter of the sphere or the natural unit of length in the hyperbolic plane, but they fail for large splines. Therefore we need to find a better method for hyperbolic splines.

Buss and Fillmore's weighted average method in [1] provides a possible solution to hyperbolic splines. The research of this thesis is to develop a method for spline curves in hyperbolic space using the weighted average method.

2 Hyperbolic Geometry

In this chapter, the basic concepts in hyperbolic geometry are explained in order for readers to fully understand this research. Lath's thesis [3] has more detailed information on hyperbolic geometry. An interested reader could refer to his paper for it.

2.1 Basic Definitions

Model

A model for a mathematical system is an interpretation of the primitive terms that makes the postulates of that system true statements. A “point”, a “line”, “lies between”, “is parallel to” are examples of primitive terms in a model for a geometry. One way to show the consistency of a geometry is to formulate a model for it.

Point

A “point” in real 3-dimensional space is represented by a triplet of real numbers, such as $\mathcal{X} = (x, y, z)$. Such a triplet can also represent a vector in 3-space. This vector can be visualized as an arrow from the origin $\mathcal{O} = (0, 0, 0)$ to \mathcal{X} . Thus the terms “vector” and “point” are interchangeable.

Spline Curves

Spline curves are curves that are specified by a small set of parameters and which have nice smoothness properties, such as having continuous k th-order derivatives. They are widely used in drafting and computer-aided manufacturing to specify curves and surfaces.

Euclidean Inner Product

Consider two vectors $\mathcal{X}_1 = (x_1, y_1, z_1)$ and $\mathcal{X}_2 = (x_2, y_2, z_2)$.

Their Euclidean inner product is defined as:

$$\mathcal{X}_1 \cdot \mathcal{X}_2 = x_1x_2 + y_1y_2 + z_1z_2.$$

Hyperbolic Inner Product

Consider two vectors $\mathcal{X}_1 = (x_1, y_1, z_1)$ and $\mathcal{X}_2 = (x_2, y_2, z_2)$.

Their hyperbolic inner product is defined as:

$$\langle \mathcal{X}_1, \mathcal{X}_2 \rangle = x_1x_2 + y_1y_2 - z_1z_2.$$

h-orthogonal

Two vectors \mathcal{X} and \mathcal{Y} are said to be h-orthogonal if $\langle \mathcal{X}, \mathcal{Y} \rangle = 0$.

h-normal

A vector \mathcal{A} is an h-normal of a plane if \mathcal{A} is h-orthogonal to any point that lies on this plane.

2.2 Models

The Euclidean plane and the sphere can be isometrically embedded in 3-dimensional Euclidean space, but the hyperbolic plane doesn't have that property. Therefore, all models of hyperbolic geometry in Euclidean 3-space have to distort distance. There are several models for hyperbolic geometry. For example, for the finite models, there are the Poincaré disk model and the Klein-Beltrami disk model, and for the infinite models, there are the Weierstrass model and the Poincaré upper-half plane model.

The research here uses the Weierstrass model and the Poincaré disk model, thus we give a brief introduction to those two models. We will give the interpretations of the primitive terms of “point”, “line”, “lies on” and “between” given by those two models. We use the symbol R^2 and R^3 to represent the Euclidean 2-space and 3-space.

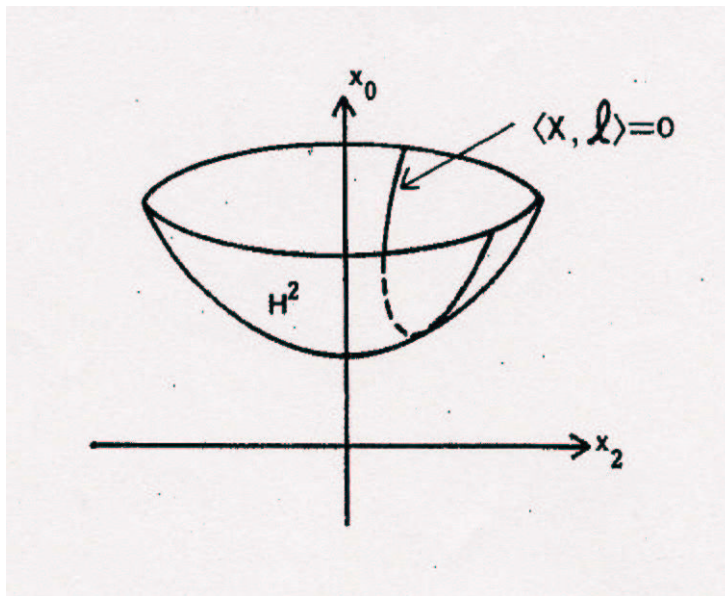


Figure 1: A “line” in the Weierstrass Model[5]

2.2.1 The Weierstrass Model

The Weierstrass model is a model which is constructed to prove consistency of hyperbolic geometry on a certain surface in Euclidean 3-space.

A “point” is a point (or vector) $\mathcal{X} = (x, y, z)$ of R^3 such that $\langle \mathcal{X}, \mathcal{X} \rangle = -k^2$, where $z > 0$ and k is a measure of the curvature of the hyperbolic plane. All such points comprise the “upper sheet” of a two sheeted hyperboloid. We call this sheet H^2 .

A “line” is the intersection of H^2 with a plane through the origin of R^3 . Such an intersection is one branch of a hyperbola (Figure 1).

A plane through the origin is given by equation $\langle X, l \rangle = 0$, where l is an h-normal of the plane. There is a vector l such that $\langle l, l \rangle = k^2$, which means that l is a point of the single-sheeted hyperboloid with the equation $\langle X, X \rangle = k^2$. Therefore, “lines” are in one-to-one correspondence with the vector l on $\langle X, X \rangle = k^2$. Thus

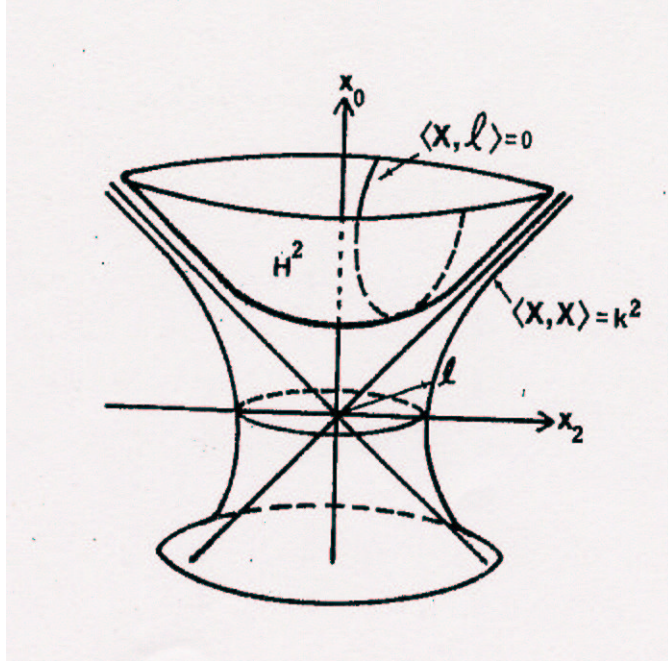


Figure 2: The Weierstrass Model[5]

we can refer to a “line” by l or giving its equation $\langle X, l \rangle = 0$.

A point \mathcal{X} of H^2 “lies on” the line l if and only if $\langle \mathcal{X}, l \rangle = 0$.

Given three distinct points A, B , and C on a line, we say C “lies between” A and B if when the line is traversed in either direction, the three points are encountered in the order ACB or BCA .

Now let’s define the distance in H^2 .

The Lobachevskian distance between two points P and Q of H^2 is the unique non-negative number $d = d_{PQ}$ satisfying

$$\cosh \frac{d}{k} = -\frac{1}{k^2} \langle P, Q \rangle .$$

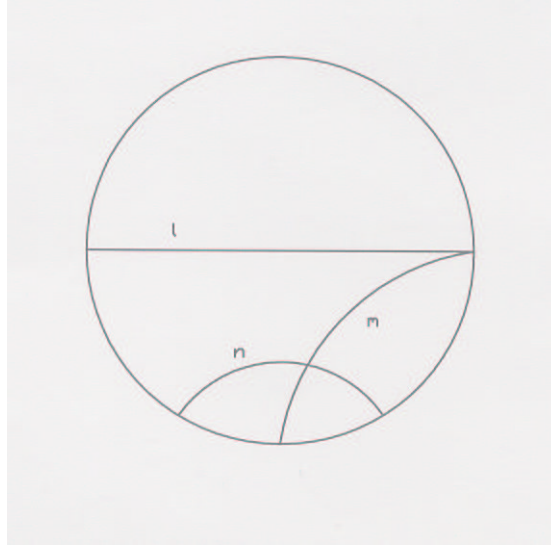


Figure 3: The Poincaré Disk Model[3]

2.2.2 The Poincaré Disk Model

The Poincaré disk model is a conformal model which means that the hyperbolic measure of the angles is the same as their Euclidean measure. However, distance is distorted.

A “point” in the Poincaré disk model is a point $X = (x, y)$ in R^2 , such that $x^2 + y^2 < 1$. Such points comprise the interior of a circle in R^2 with center at the origin and unit radius. The points on the circumference of this circle are not included in the model. We will call this circle D or “the Poincaré disk”.

A “line” is a set of interior points of D that constitute an arc of a circle that is orthogonal to D . Open diameters of D (arcs of a circle of infinite radius and obviously intersect D orthogonally) are also “lines”.

The terms “lies on” and “lies between” have the same meanings as in the Euclidean case.

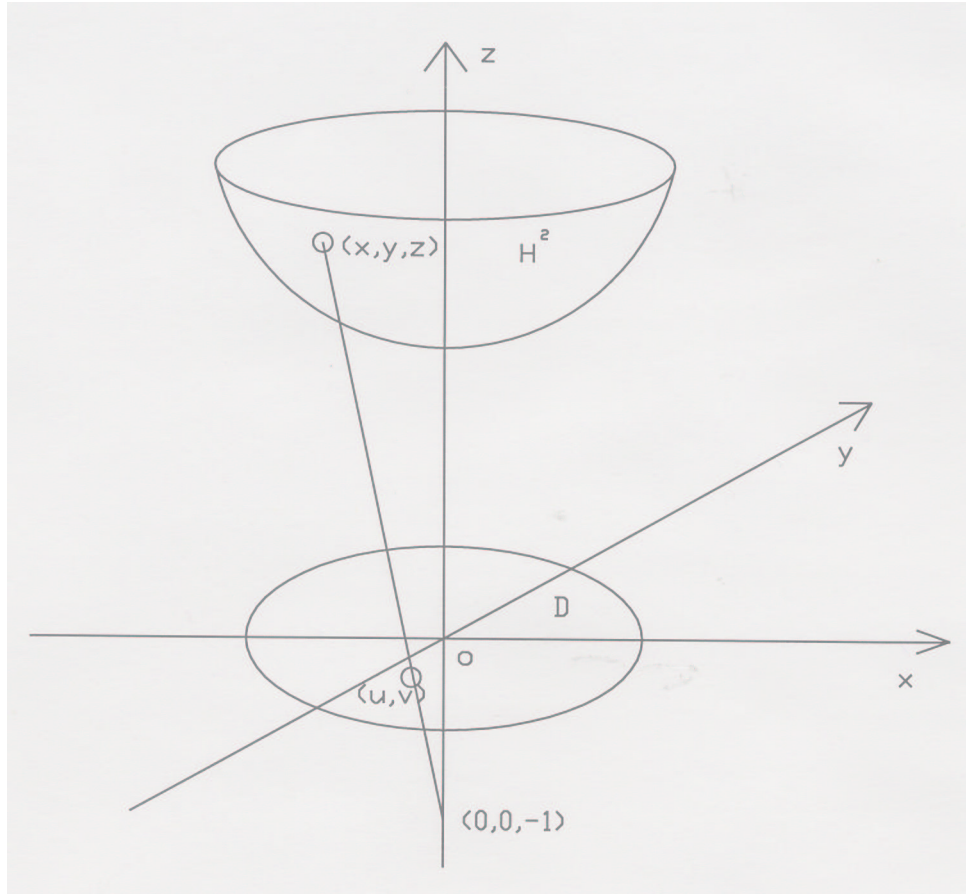


Figure 4: Isomorphism between the Weierstrass (H^2) and the Poincaré Disk Model (D)[3]

In Figure 3, there are three lines in the Poincaré disk model. We say that l and n are divergently parallel, l and m are asymptotically parallel, and m and n are intersecting.

2.2.3 Isomorphism between the Weierstrass and the Poincaré Disk Model

The Weierstrass model is where we do all our calculations and the Poincaré disk model is where we display the visual results. The isomorphism between those two models is as follows:

Pick a point arbitrarily on the Weierstrass model with coordinates (x, y, z) . Its corresponding point on the Poincaré disk model has coordinates (u, v) . The relationships between those two points are

$$u = \frac{x}{1+z}, \tag{1}$$

$$v = \frac{y}{1+z},$$

and

$$x = \frac{2u}{1-u^2-v^2},$$

$$y = \frac{2v}{1-u^2-v^2}, \tag{2}$$

$$z = \frac{1+u^2+v^2}{1-u^2-v^2}.$$

3 Parametric Curves

3.1 Representations of a Curve in 3-Dimensional Euclidean Space

There are several ways to represent a curve in 3-dimensional Euclidean space mathematically, as described in *Computer Graphics: Principles and Practice* [2].

In the first way, “the function form”, we express y and z as explicit functions of x , so $y = f(x)$ and $z = g(x)$. It has some disadvantages. (1) For a single x , it is impossible to get multiple values of y and z . Therefore it is not a good way to represent curves like circles and ellipses. (2) This kind of definition is not rotationally invariant. Much calculation is required to describe a rotated version and it may be necessary to break a curve into many segments. (3) A slope of infinity is difficult to represent, therefore describing a curve with a vertical tangent is difficult.

In the second way, we use an implicit equation of the form $f(x, y, z) = 0$ to represent curves. This method may give more solutions than we want. It also has the difficulty of determining if two intersecting curves are tangent at the point of intersection.

In computer graphics, people tend to use parametric curves because of their advantages over the functional and implicit forms. It overcomes the difficulty of representing curves with vertical slopes by replacing the use of geometric slopes with parametric tangent vectors. It also makes it possible to have multiple values of y and z for a given x . It provides other advantages too.

3.2 Parametric Cubic Curves

Parametric cubic curves are the most used parametric forms of representing a curve in 3-dimensional Euclidean Space. It is because lower-degree polynomials give little flexibility in controlling the shape of the curves, and higher-degree polynomials may introduce unwanted wiggles and are computationally more expensive.

The general form of a 3-dimensional parametric cubic curve segment $Q(t) = [x(t) \ y(t) \ z(t)]$ is

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x, \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y, \\ z(t) &= a_z t^3 + b_z t^2 + c_z t + d_z, \quad 0 \leq t \leq 1. \end{aligned} \tag{3}$$

The curve $Q(t)$ can be represented as the product of two matrices, $Q(t) = T \cdot C$, where $T = [t^3, t^2, t, 1]$ and the coefficient matrix C can be rewritten as $C = M \cdot G$, where M is a 4×4 basis matrix, and G is a 4-element column matrix, in which the elements are geometric constraints.

$$\begin{aligned} Q(t) &= [x(t) \ y(t) \ z(t)] \\ &= T \cdot M \cdot G \\ &= (t^3 \ t^2 \ t \ 1) \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{pmatrix} \begin{pmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{pmatrix}. \end{aligned} \tag{4}$$

The elements of M and G are constants, therefore the product is just cubic polynomials in t .

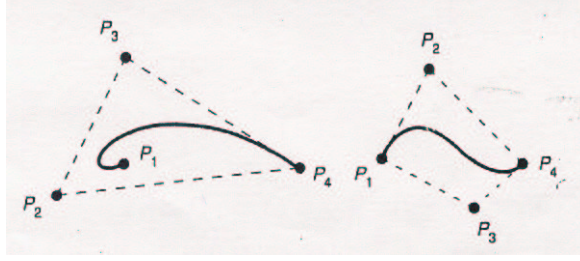


Figure 5: Two Bézier Curves[2]

Expanding $x(t)$, we get:

$$\begin{aligned}
 x(t) = & (t^3 m_{11} + t^2 m_{21} + t m_{31} + m_{41}) g_{1x} \\
 & + (t^3 m_{12} + t^2 m_{22} + t m_{32} + m_{42}) g_{2x} \\
 & + (t^3 m_{13} + t^2 m_{23} + t m_{33} + m_{43}) g_{3x} \\
 & + (t^3 m_{14} + t^2 m_{24} + t m_{34} + m_{44}) g_{4x}.
 \end{aligned} \tag{5}$$

From equation (5), we can see that the curve is a weighted sum of the elements of the geometry matrix.

3.3 Bézier Curves

Bézier curve (named after mathematician Pierre Bézier) is a form of the cubic polynomial curve segment. It specifies four control points in which the two intermediate points are not on the curve.

Euclidean Bézier curves are represented as

$$\begin{aligned}
 Q(t) &= T \cdot M_B \cdot G_B \\
 &= (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t) P_3 + t^3 P_4,
 \end{aligned} \tag{6}$$

where M_B is Bézier basis matrix, G_B is Bézier geometry vector, P_1, P_4 are two end points on the curve, and P_2, P_3 are two intermediate points which are not on the curve. The tangent vectors at the two end points are determined by vectors $P_1 P_2$ and $P_3 P_4$.

We can represent Bézier curves as the weighted average

$$Q(t) = w_1P_1 + w_2P_2 + w_3P_3 + w_4P_4, \quad (7)$$

where the weights are $w_1 = (1 - t)^3$, $w_2 = 3t(1 - t)^2$, $w_3 = 3t^2(1 - t)$, $w_4 = t^3$.

In this research, we use Bézier curves as the form of hyperbolic splines.

4 The Weighted Average Algorithm for Hyperbolic Splines

Over the years, some algorithms have been developed for splines on the sphere. In 2001, mathematicians Samuel Buss and Jay Fillmore developed a weighted average algorithm for spherical splines [1]. Their algorithm overcomes the shortcomings of the previous methods in this area and provides a possible solution for splines in hyperbolic space.

Buss and Fillmore’s method for calculating the weighted averages on spheres is based on “*least squares minimization that respects spherical distance*”. They proved the existence and uniqueness of the weighted averages and also gave two iterative algorithms, one with linear convergence rate, the other with quadratic convergence rate. An interested reader can refer to [1] for their entire work.

4.1 Hyperbolic Isometric Transformations

We think the weighted average algorithm could be a solution for hyperbolic splines. The goal of this research is to develop a weighted average algorithm with linear convergence rate for hyperbolic splines which is similar to the one for spherical splines.

In order to develop the algorithm, first we need to define some basic ideas and give some transformation matrices.

We use Bézier curves as the form of our hyperbolic splines, therefore we need four control points $p_i = (x_i, y_i, z_i)$ on the hyperboloid, where $i = 1, \dots, 4$. Let w_1, \dots, w_4 be weights of Bézier curves, where $w_i \geq 0$ and $\sum_{i=1}^4 w_i = 1$.

The Euclidean weighted average q is

$$q = \begin{pmatrix} x_q \\ y_q \\ z_q \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^4 w_i x_i \\ \sum_{i=1}^4 w_i y_i \\ \sum_{i=1}^4 w_i z_i \end{pmatrix} = \sum_{i=1}^4 w_i p_i.$$

The length of q is

$$\|q\| = \left\| \sum_{i=1}^4 w_i p_i \right\| = \sqrt{z_q^2 - x_q^2 - y_q^2}.$$

In order to transfer points between the Weierstrass hyperboloid and the tangent plane to it at $q = (0, 0, 1)$, we need two functions $exp_q()$ and $l_q()$.

The “exponential” function $exp_q()$ is a function which maps a point p with coordinates (u, v) on the tangent plane to the point $exp_q(p) = (x, y, z)$ on the hyperboloid, where

$$x = u \cdot \frac{\sinh r}{r}, \quad y = v \cdot \frac{\sinh r}{r}, \quad z = \cosh r,$$

and

$$r = \sqrt{u^2 + v^2}.$$

The “logarithm” function $l_q()$ is the inverse of $exp_q()$. It maps a point p with coordinates (x, y, z) on the hyperboloid to the point $l_q(p) = (u, v)$ on the tangent plane, where

$$u = x \cdot \frac{\theta}{\sinh \theta}, \quad v = y \cdot \frac{\theta}{\sinh \theta},$$

and

$$\theta = \cosh^{-1}(z).$$

It is easy to see that $exp_q(l_q(p)) = p$.

Now we introduce some transformation matrices. The goal is to transfer an arbitrary point (x, y, z) on the hyperboloid to the “origin” $o = (0, 0, 1)$.

Let p be an arbitrary point with coordinates (x, y, z) on the hyperboloid . Let p' be the projection point of p on the xz -plane; p' has coordinates $(x, 0, z)$. Let θ be the angle between p and the xz -plane. Let d be the distance between p' and the origin o on the hyperboloid.

Let $\cos \theta = \frac{x}{r}$, $\sin \theta = \frac{y}{r}$ where $\sqrt{x^2 + y^2} = r$.

From Figure 6, we get

$$\begin{pmatrix} x \\ 0 \\ z \end{pmatrix} = \begin{pmatrix} \sinh d \\ 0 \\ \cosh d \end{pmatrix}.$$

The reflection in bisector of Figure 6 is given by

$$\begin{pmatrix} -\cosh d & 0 & \sinh d \\ 0 & 1 & 0 \\ -\sinh d & 0 & \cosh d \end{pmatrix}.$$

The reflection across the yz -plane is given by

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The translation of the point $p' = (x, 0, z)$ of the xz -plane to the origin $o = (0, 0, 1)$ is given by

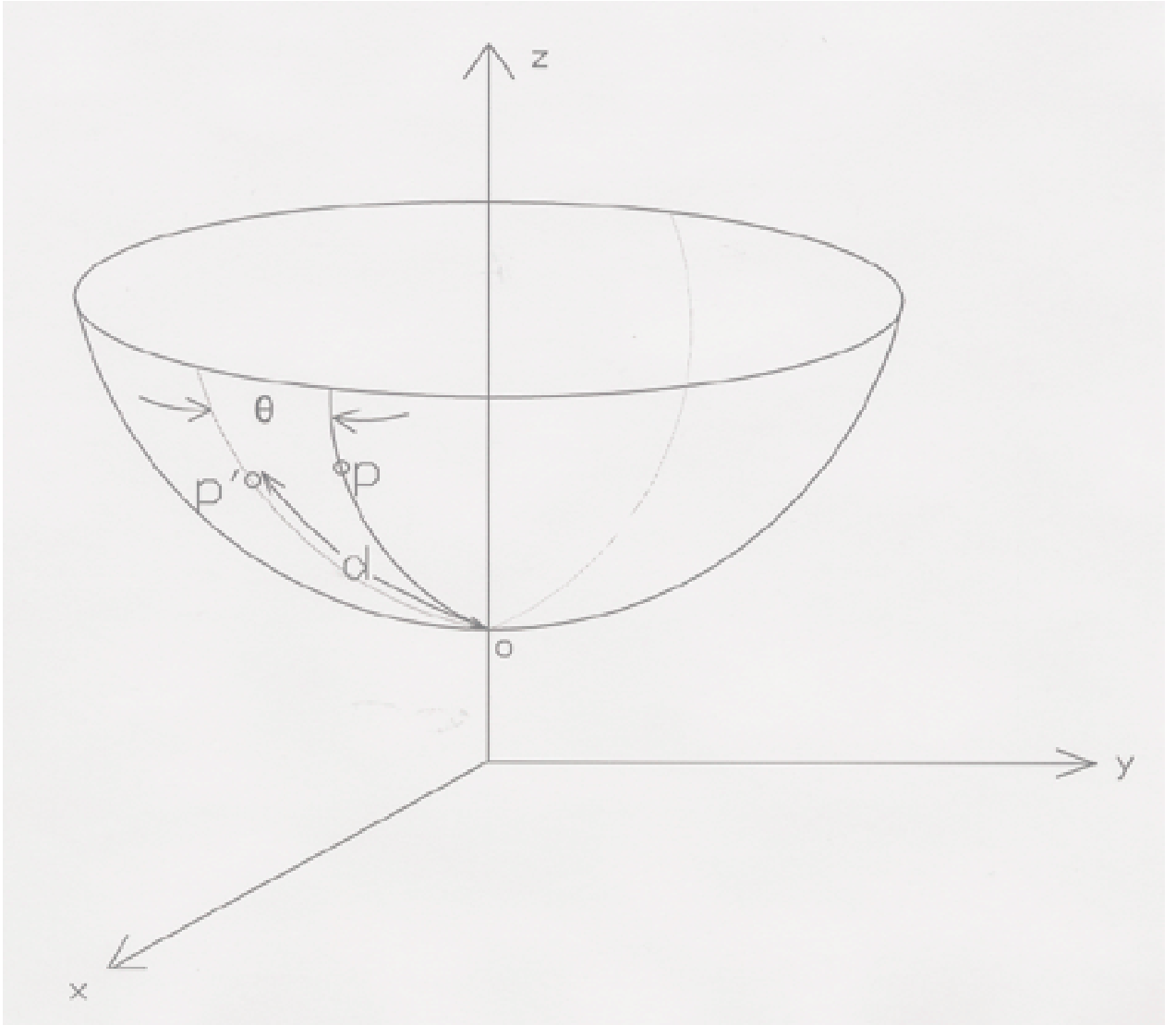


Figure 6: Point p on the Hyperboloid

$$\begin{aligned}
Trans(d) &= \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -\cosh d & 0 & \sinh d \\ 0 & 1 & 0 \\ -\sinh d & 0 & \cosh d \end{pmatrix} \\
&= \begin{pmatrix} \cosh d & 0 & -\sinh d \\ 0 & 1 & 0 \\ -\sinh d & 0 & \cosh d \end{pmatrix} \\
&= \begin{pmatrix} z & 0 & -x \\ 0 & 1 & 0 \\ -x & 0 & z \end{pmatrix}.
\end{aligned}$$

Since the translation is on the xz -plane, $y = 0$, $r = \sqrt{x^2 + y^2} = \sqrt{x^2} = x$. Therefore,

$$Trans(d) = \begin{pmatrix} z & 0 & -r \\ 0 & 1 & 0 \\ -r & 0 & z \end{pmatrix}. \tag{8}$$

The matrix for rotating by the angle of θ is given by

$$Rot(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The matrix for rotating by the angle of $-\theta$ is given by

$$Rot(-\theta) = \begin{pmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Replacing $\cos \theta$ and $\sin \theta$ with $\frac{x}{r}$ and $\frac{y}{r}$, we get

$$Rot(\theta) = \begin{pmatrix} \frac{x}{r} & -\frac{y}{r} & 0 \\ \frac{y}{r} & \frac{x}{r} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad (9)$$

$$Rot(-\theta) = \begin{pmatrix} \frac{x}{r} & \frac{y}{r} & 0 \\ -\frac{y}{r} & \frac{x}{r} & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (10)$$

To project the point p with coordinates (x, y, z) to the xz -plane to get $p' = (x, 0, z)$, apply the matrix $Rot(-\theta)$ to p ,

$$\begin{pmatrix} \frac{x}{r} & \frac{y}{r} & 0 \\ -\frac{y}{r} & \frac{x}{r} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{x^2 + y^2}{r} \\ 0 \\ z \end{pmatrix} = \begin{pmatrix} r \\ 0 \\ z \end{pmatrix}.$$

Therefore to transfer an arbitrary point p to the origin o on the hyperboloid, we can first project it onto the xz -plane to get p' by matrix $Rot(-\theta)$, then move p' to the origin o by matrix $trans(d)$, then rotate it back by matrix $Rot(\theta)$ as follows:

The translation matrix T of transferring an arbitrary point p with coordinates

(x, y, z) to the origin o is:

$$\begin{aligned}
T &= Rot(\theta) \cdot trans(d) \cdot Rot(-\theta) \\
&= \begin{pmatrix} \frac{x}{r} & -\frac{y}{r} & 0 \\ \frac{y}{r} & \frac{x}{r} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} z & 0 & -r \\ 0 & 1 & 0 \\ -r & 0 & z \end{pmatrix} \begin{pmatrix} \frac{x}{r} & \frac{y}{r} & 0 \\ -\frac{y}{r} & \frac{x}{r} & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} \frac{x^2z + y^2}{r^2} & \frac{xy(z-1)}{r^2} & -x \\ \frac{xy(z-1)}{r^2} & \frac{y^2z + x^2}{r^2} & -y \\ -x & -y & z \end{pmatrix}.
\end{aligned} \tag{11}$$

4.2 Algorithm for Hyperbolic Weighted Averages

In Buss and Fillmore's paper [1], they defined the weighted average as “the result of a least squares minimization, namely, as the point C on S^d that minimizes the value:”

$$f(C) = \frac{1}{2} \sum_i w_i \cdot dist_S(C, p_i)^2,$$

where $dist_S()$ is the length of the shortest geodesic between points C and p_i on S^d (S^d is the d -dimensional unit sphere in R^{d+1}).

They proved the existence and uniqueness properties of the weighted average in Theorem 1:

Theorem 1

Suppose the points p_1, \dots, p_n all lie in a hemisphere H of S^d , with at least one point p_i in the interior of H with $w_i \neq 0$. Then the function f has a single critical point q

in H , and this point q is the global minimum of f .

They also proved another theorem:

Theorem 2

Let q be a critical point of f , with q not antipodal to any p_i . Then

$$\sum_{i=1}^n w_i (l_q(p_i) - q) = 0.$$

The research here exchanges S^d for hyperbolic space and develops an algorithm similar to the one in [1]. It is a linear convergence rate algorithm. It iterates to search for a point q which satisfies the condition specified in Theorem 2.

Here is the weighted average algorithm for hyperbolic splines.

Algorithm

Inputs: Points p_1, p_2, p_3, p_4 on the hyperboloid.

Output: A set of weighted averages which constitute the spline specified by points p_1, p_2, p_3, p_4 .

Main Loop:

For $t = 0, 1/16, 2/16, \dots, 1$

$$w_1 = (1 - t)^3,$$

$$w_2 = 3t(1 - t)^2,$$

$$w_3 = 3t^2(1 - t),$$

$$w_4 = t^3.$$

Initialization: Set $q = \frac{\sum_{i=1}^4 w_i p_i}{\| \sum_{i=1}^4 w_i p_i \|}$.

Inside Loop:

For $i = 1, \dots, 4$

$$p_i = Tp_i.$$

Set $p_i^* = l_q(p_i)$.

Set $u = \sum_{i=1}^4 w_i p_i^*$.

Set $q = exp_q(u)$.

Get the new T from the new q .

If $\| u \|$ is small enough, output q and halt.

Otherwise continue looping.

$$q = T^{-1}q.$$

The algorithm calculates the data for the shape of a hyperbolic spline if the four control points are provided. The first and fourth points are end points of the spline, while the second and third points are intermediate points which control the shape of the spline. In the algorithm, t is a parameter which is used to calculate the weights of Bézier curve. We let t be in the range of $[0, 1]$, therefore at $t = 0$, the weighted average is just the first control point, and at $t = 1$, the weighted average is just the fourth control point. When t is changing between 0 and 1, as in the outside loop in the algorithm, all the weighted averages calculated by the algorithm constitute the spline specified by those four control points.

In the algorithm above, first we give the weighted average an initial value. Here we let it be the normalized Euclidean weighted average

$$q = \frac{\sum_{i=1}^4 w_i p_i}{\| \sum_{i=1}^4 w_i p_i \|}.$$

We need to get the translation matrix T of the weighted average q which translates q to the origin. This translation makes the calculations easier because we then can do all the calculations on the tangent plane of q . The four control points p_i get translated as well, therefore we need to apply T to those four control points too.

Then, the algorithm maps the transferred control points to the tangent plane at q by function $l_q(p_i)$. We calculate the Euclidean weighted average u in the tangent plane, then map it back to the hyperboloid by function $exp_q(u)$. According to [1], the vector u is actually the negative of the gradient $\nabla f(q)$ of f at q . Therefore, the inside loop moves q in the direction of the steepest descent. This algorithm gives a linear convergence rate which is what we observed in our experiments.

4.3 Transformation between the Visual Plane and the Hyperboloid

The hyperbolic splines are displayed on a 2-dimensional plane like a computer monitor screen or a piece of paper, but the whole calculations for the weighted averages is on the 3-dimensional hyperboloid. Therefore we need to use different hyperbolic models. The Poincaré disk model can be the 2-dimensional plane on which we visually display the splines. The Weierstrass model can be the model for the hyperboloid on which we do all the calculations. Therefore the transformation between the visual plane and the hyperboloid is just the transformation between the Poincaré disk model and the Weierstrass model which is defined in section 2.2.3.

If the algorithm of this thesis is used by a computer program, transformation between the visual plane and the hyperboloid becomes inevitable. The user friendly interface of the computer program may have the function which allows the user to click four points on the screen, then draw a hyperbolic spline defined by those four points. The points the user clicked on the screen are 2-dimensional. We need to transfer them to 3-dimensional points by equation (2) in order for our algorithm to use. After calculation, the algorithm provides results in 3-dimensional. We need to use equation (1) to transfer them back to 2-dimensional data and draw the spline on the screen.

4.4 *C* Program to Calculate Hyperbolic Spline Data

Using the algorithm discussed in this research, a *C* program called `spline.c` is written to calculate the spline for any given four control points. The first and fourth points are the end points of the spline, and the two intermediate points control the shape of the spline.

This program can be used by itself if the user provides four control points and defines the output format. It can also be used with Professor Douglas Dunham's hyperbolic pattern generating program in which there is a button to draw hyperbolic splines. The user can click the button, then click any four points on the canvas within the Poincaré disk. The hyperbolic spline defined by those four points will be generated on the canvas. The research in this thesis fills the void of generating hyperbolic spline curves for his program.

The source code of the *C* program can be found in the Appendix.

5 Results and Analyses

We have implemented the algorithm in Professor Dunham's hyperbolic pattern generation program. His program creates a pattern composed of hyperbolically congruent copies of a basic subpattern. This section gives some graphic results of hyperbolic splines from his program.

In order to see the difference between hyperbolic and Euclidean splines, we let the program draw a subpattern which consists of four lines defined by four control points given by the user. For the four points, we generate a hyperbolic spline, a Euclidean spline, a Euclidean polyline which connects four points in order, and a hyperbolic polyline which connects the points in order.

5.1 Splines and Their Tangents at Ends

Figures 7 to 9 give graphic results of one hyperbolic spline and its corresponding Euclidean spline defined by four given control points. Figure 7 shows four different kinds of lines, a hyperbolic spline (red), a Euclidean spline (green), a hyperbolic polyline (blue) and a Euclidean polyline (magenta). In Figure 8, only the hyperbolic spline (red) and the hyperbolic polyline (blue) are shown. In Figure 9, only the Euclidean spline (green) and the Euclidean polyline (magenta) are shown. From the graphs, we can see that the hyperbolic spline is tangent to the hyperbolic polyline at the first and fourth point (Figure 8), while the Euclidean spline is tangent to the Euclidean polyline at those two points (Figure 9).

Figures 10 to 21 give similar graphic results for four other hyperbolic splines.

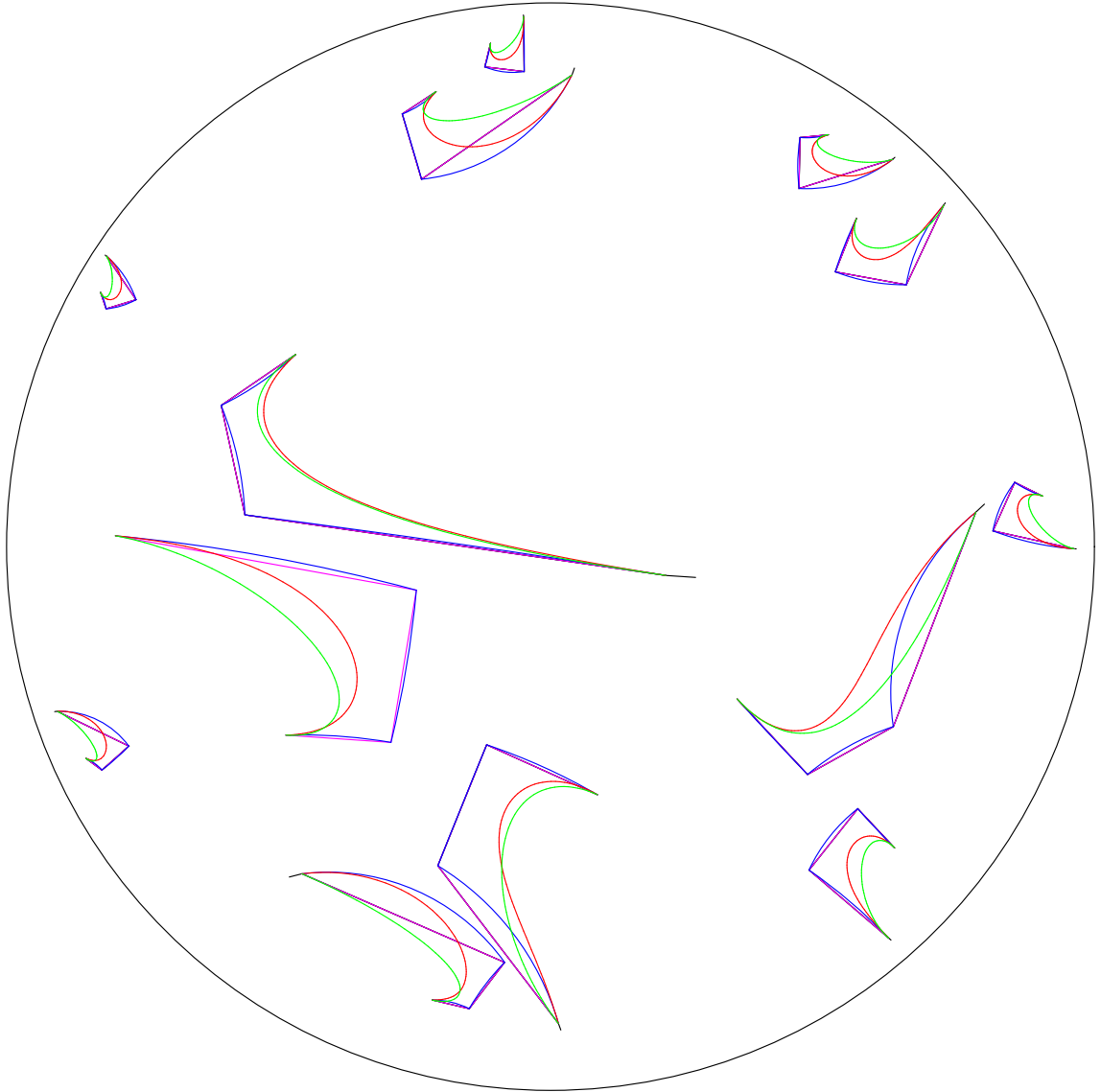


Figure 7: Spline 1

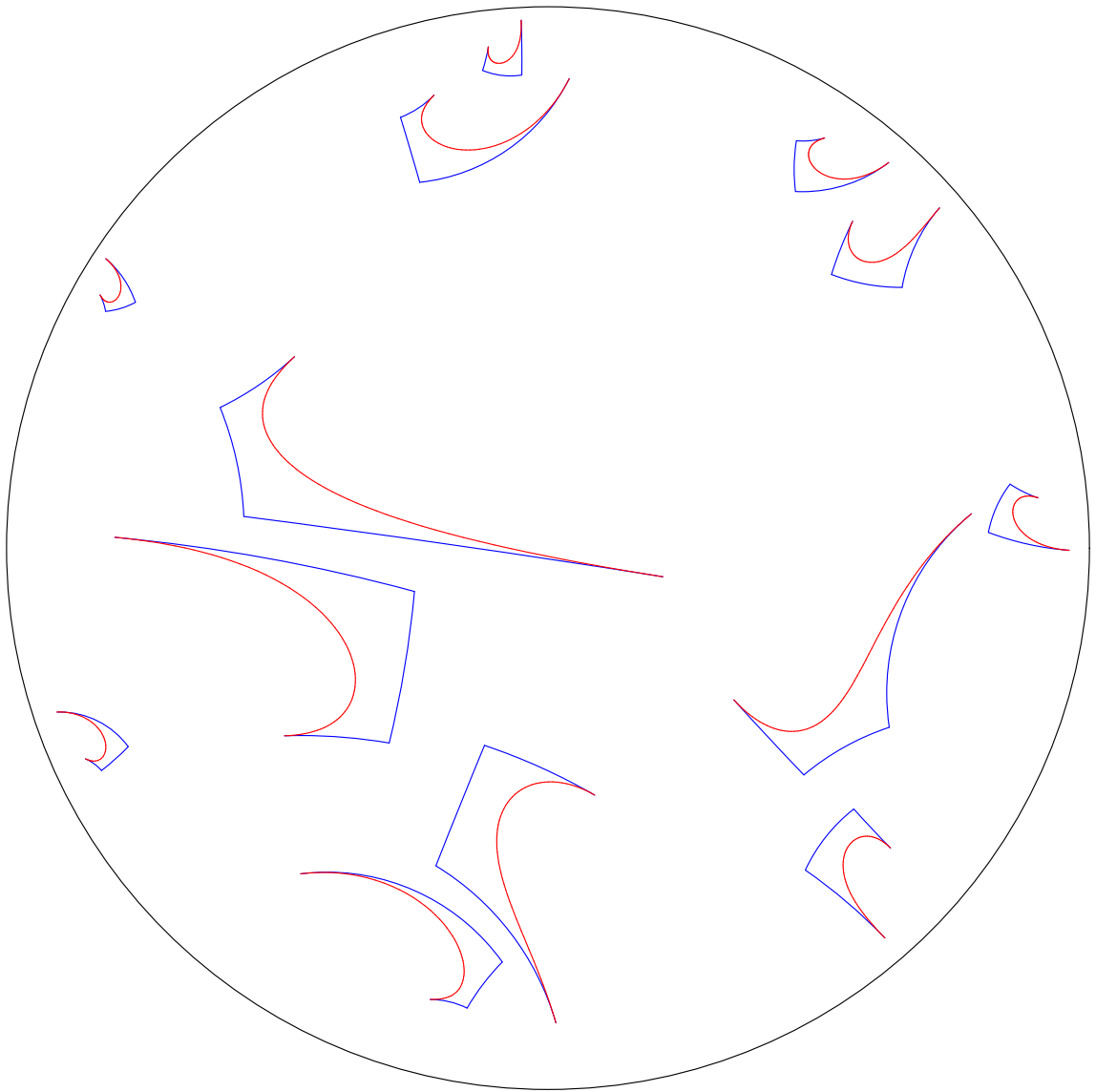


Figure 8: Hyperbolic Spline (red) and Hyperbolic Polyline (blue)

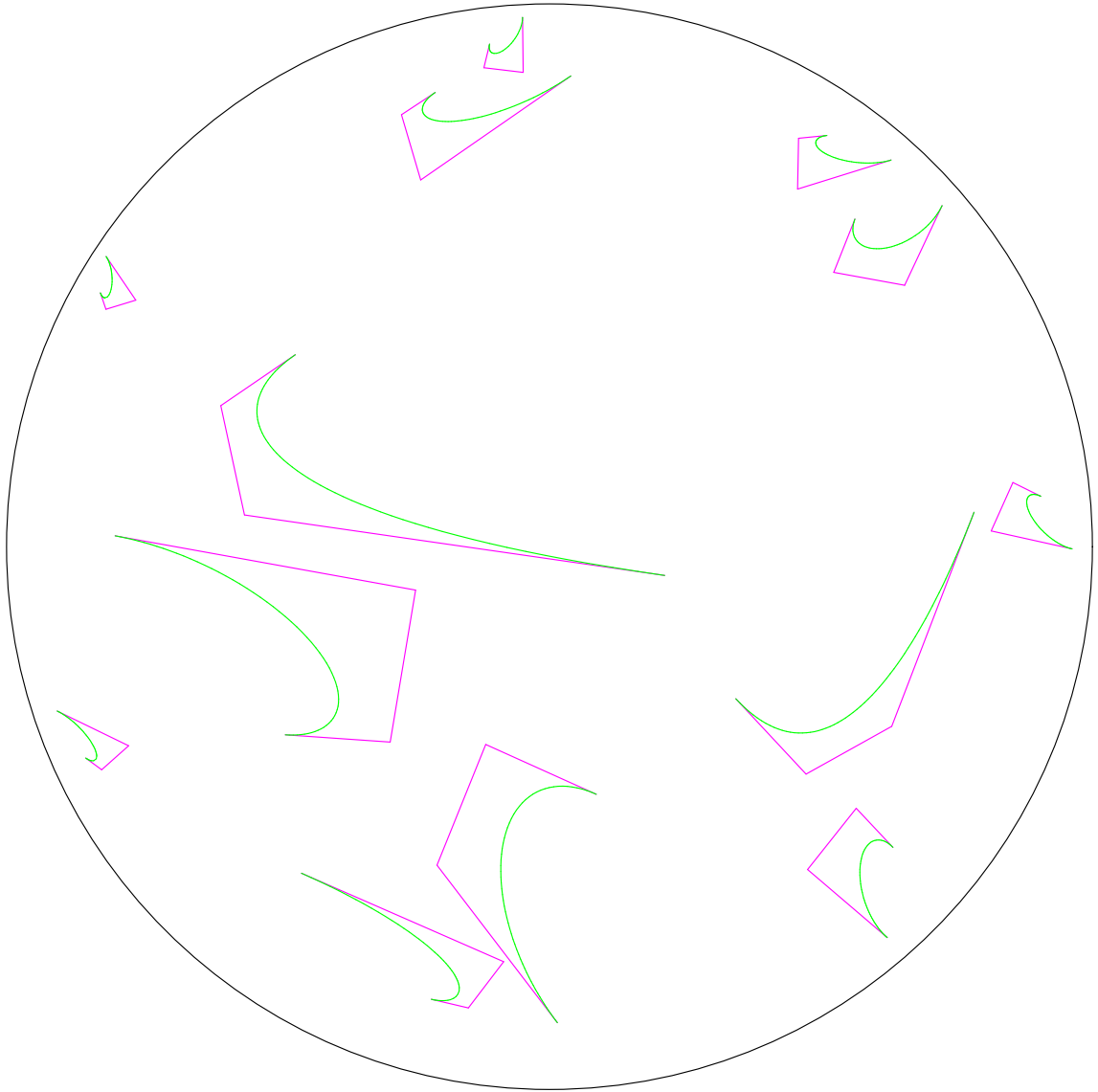


Figure 9: Euclidean Spline (green) and Euclidean Polyline (magenta)

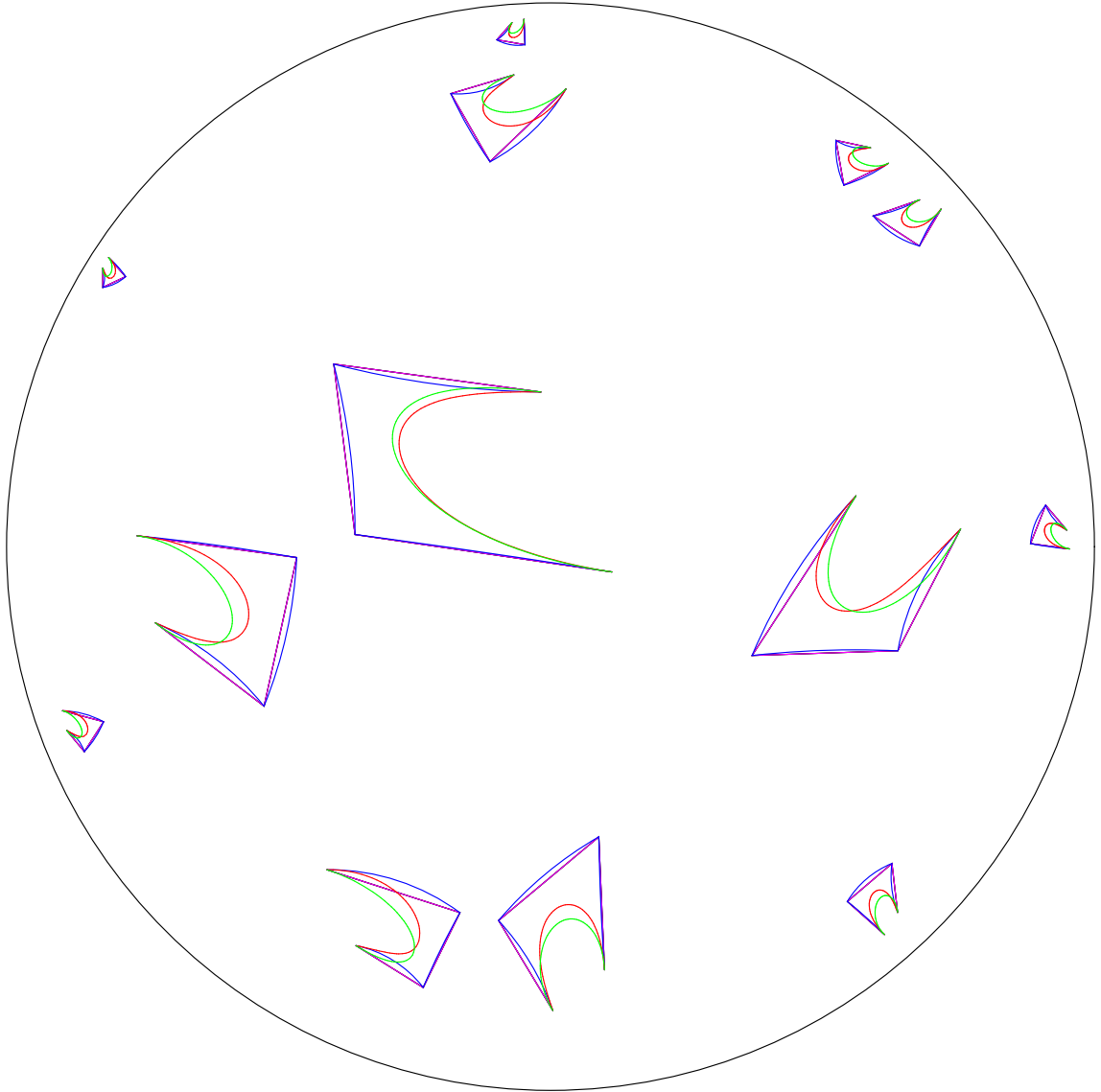


Figure 10: Spline 2

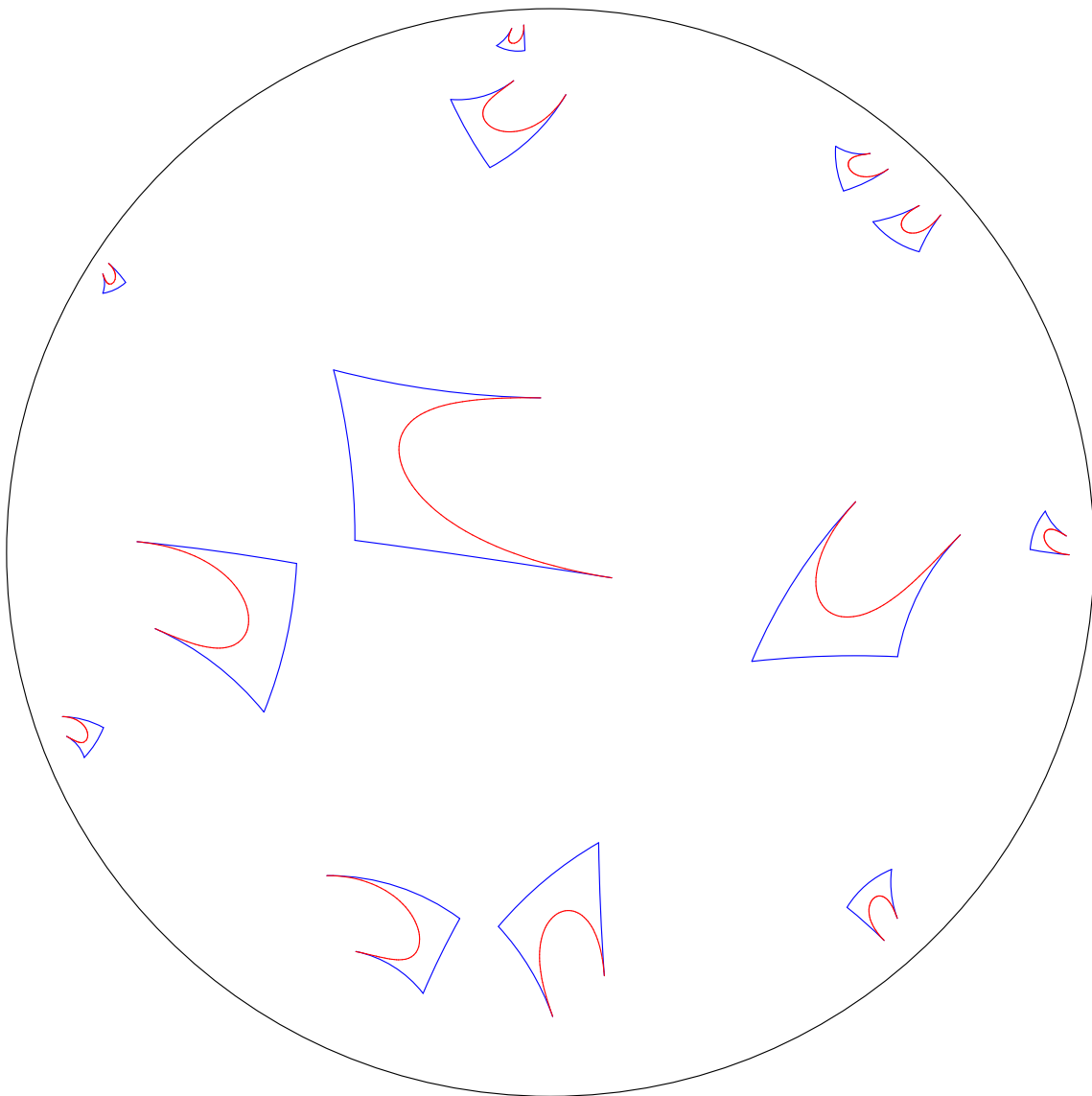


Figure 11: Hyperbolic Spline (red) and Hyperbolic Polyline (blue)

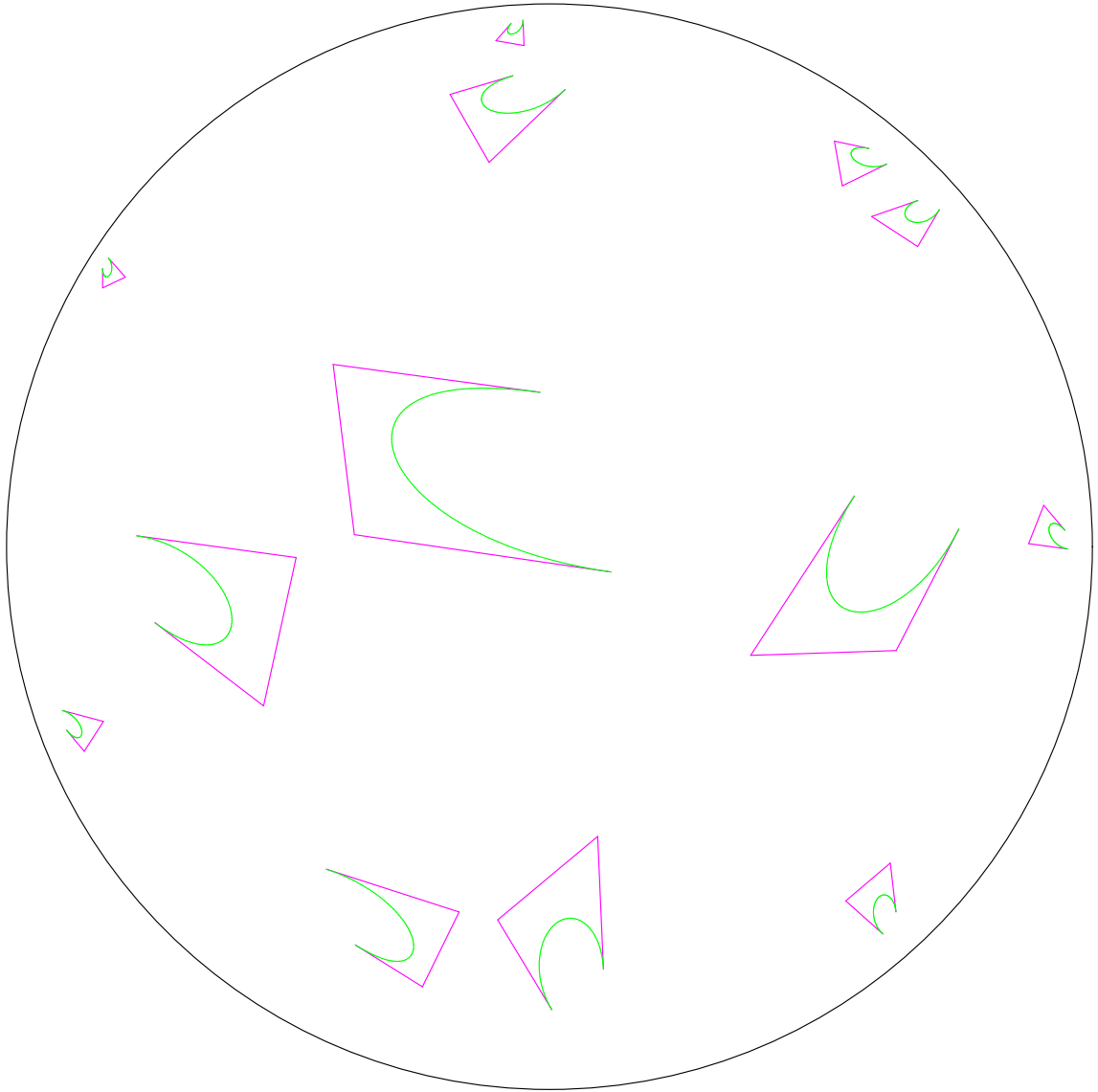


Figure 12: Euclidean Spline (green) and Euclidean Polyline (magenta)

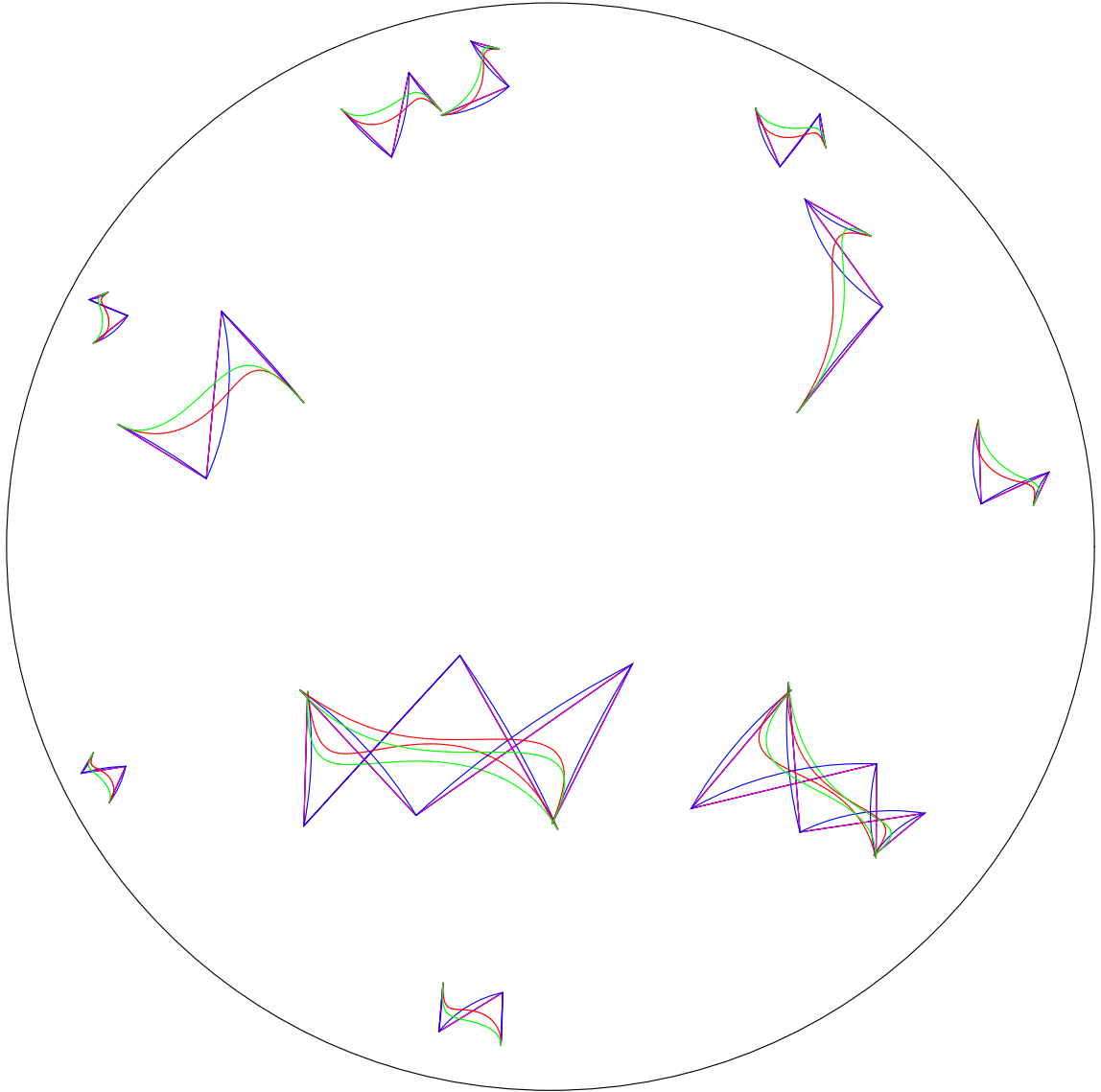


Figure 13: Spline 3

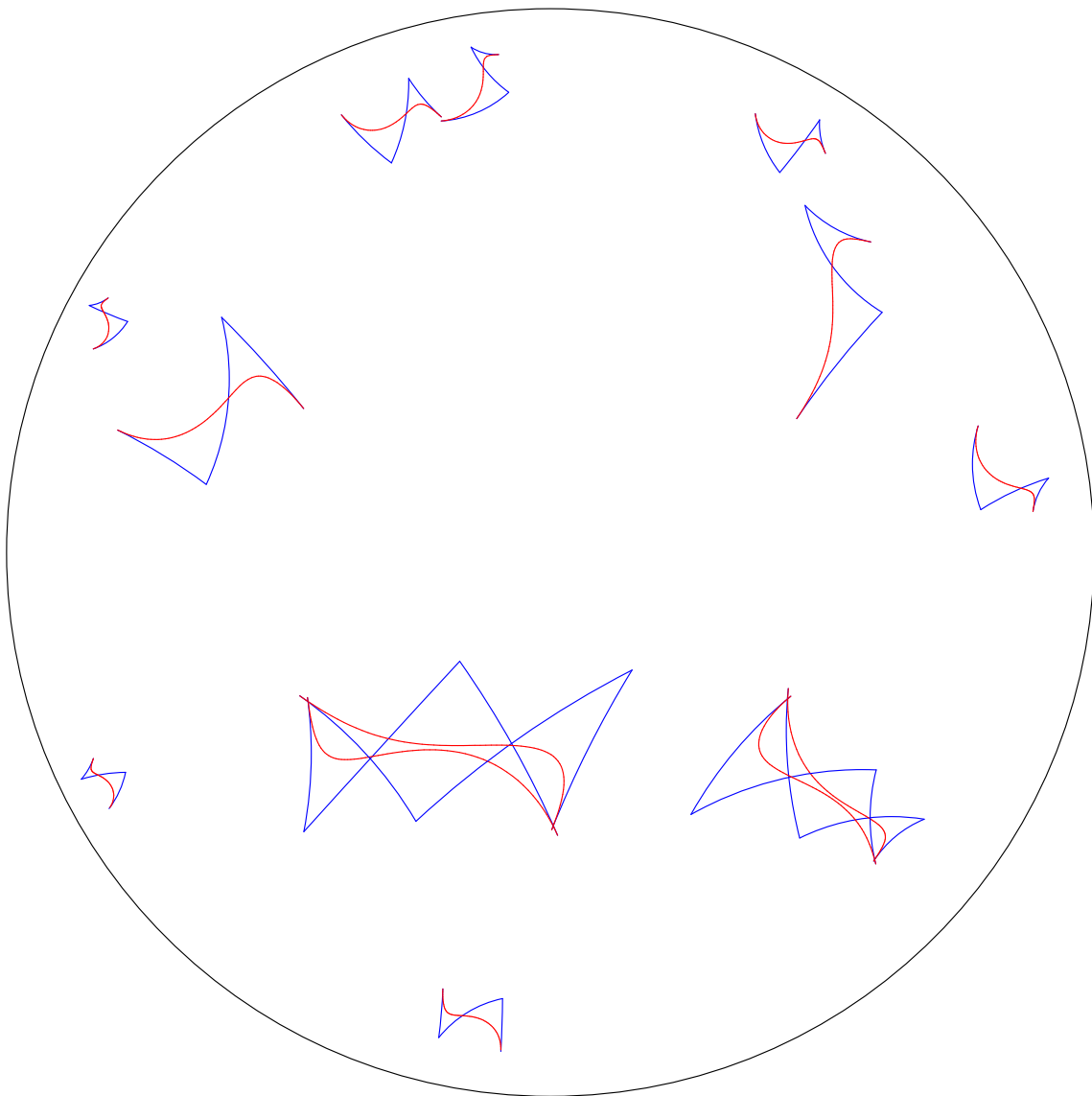


Figure 14: Hyperbolic Spline (red) and Hyperbolic Polyline (blue)

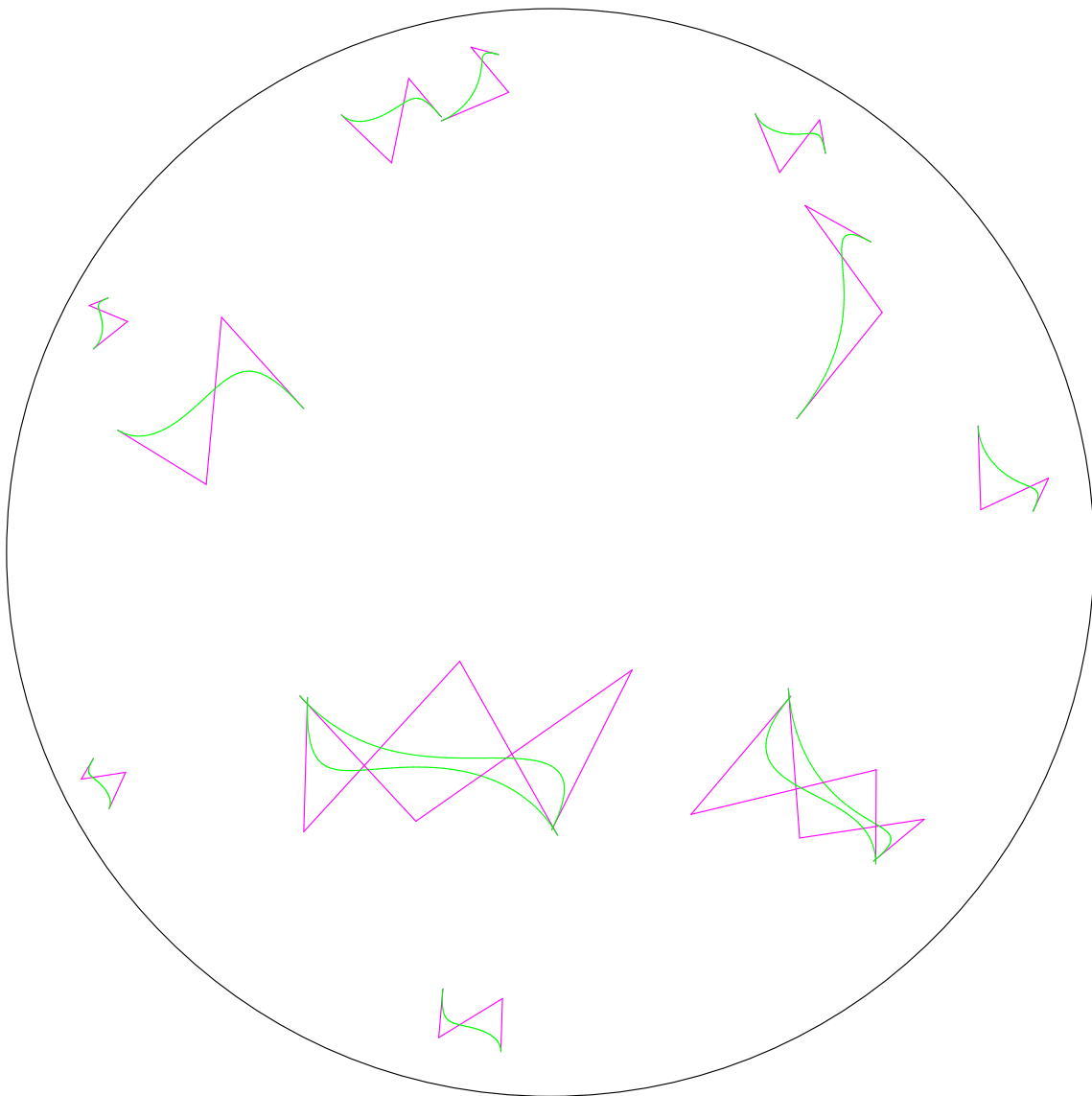


Figure 15: Euclidean Spline (green) and Euclidean Polyline (magenta)

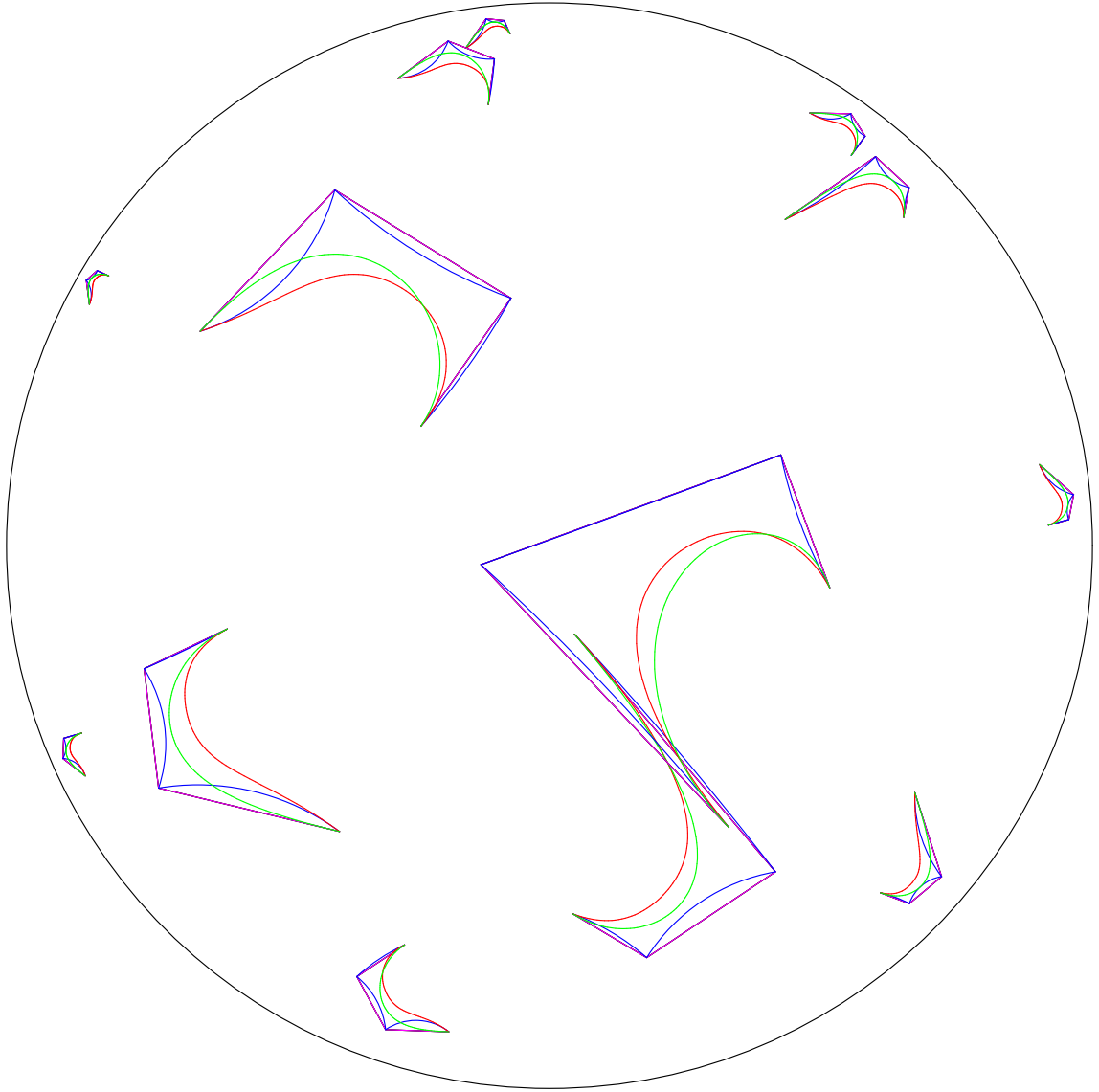


Figure 16: Spline 4

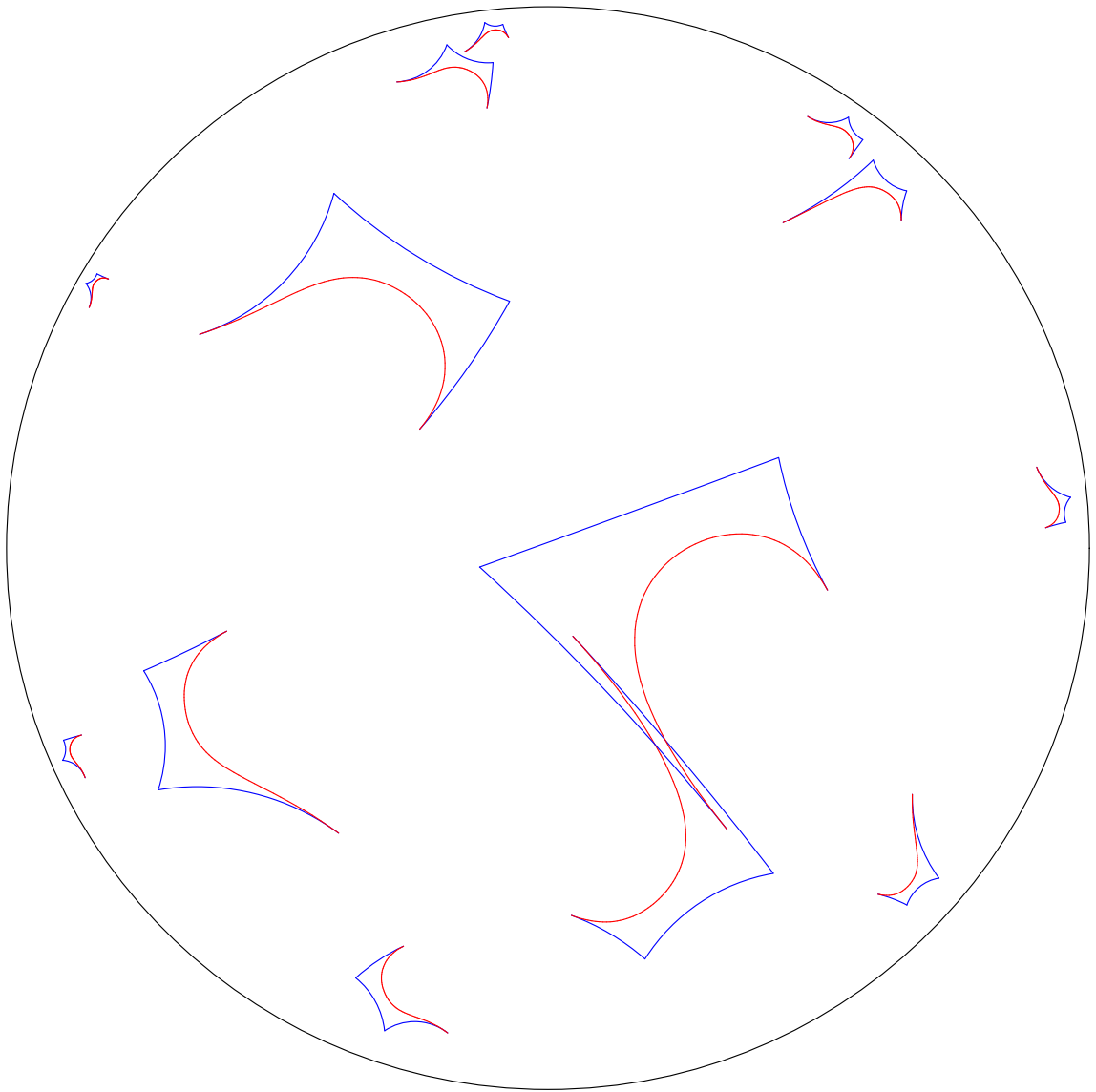


Figure 17: Hyperbolic Spline (red) and Hyperbolic Polyline (blue)

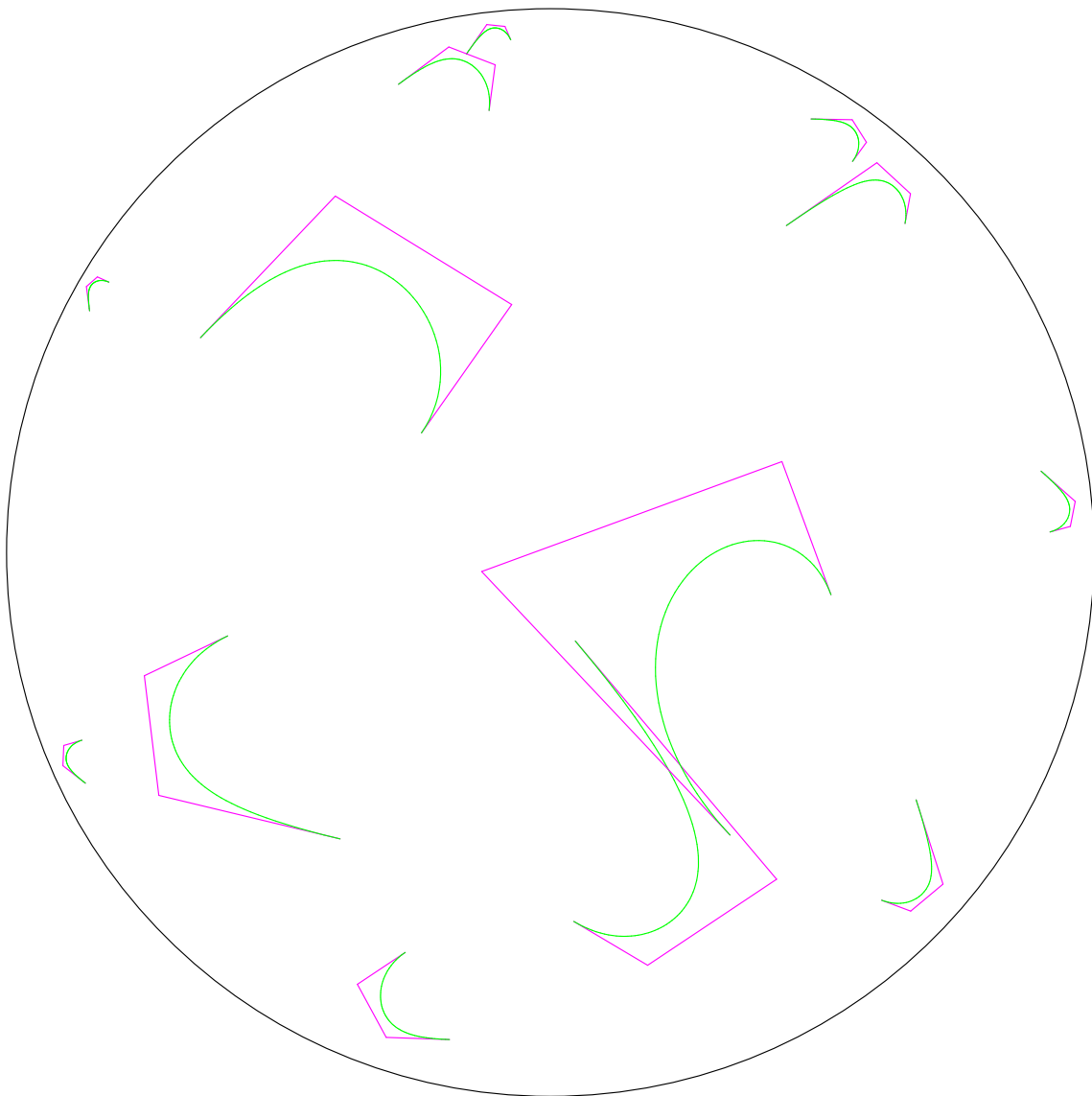


Figure 18: Euclidean Spline (green) and Euclidean Polyline (magenta)

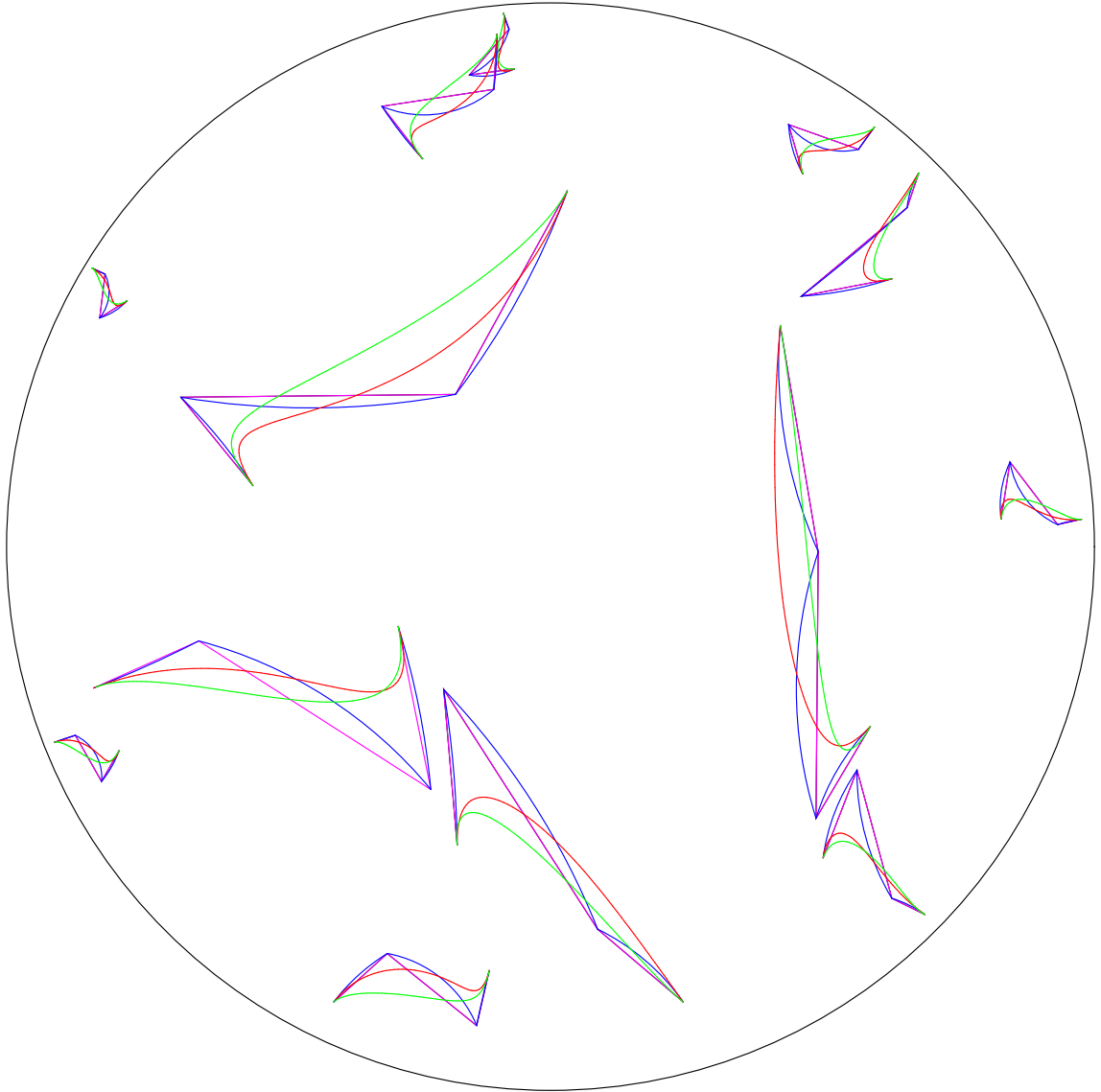


Figure 19: Spline 5

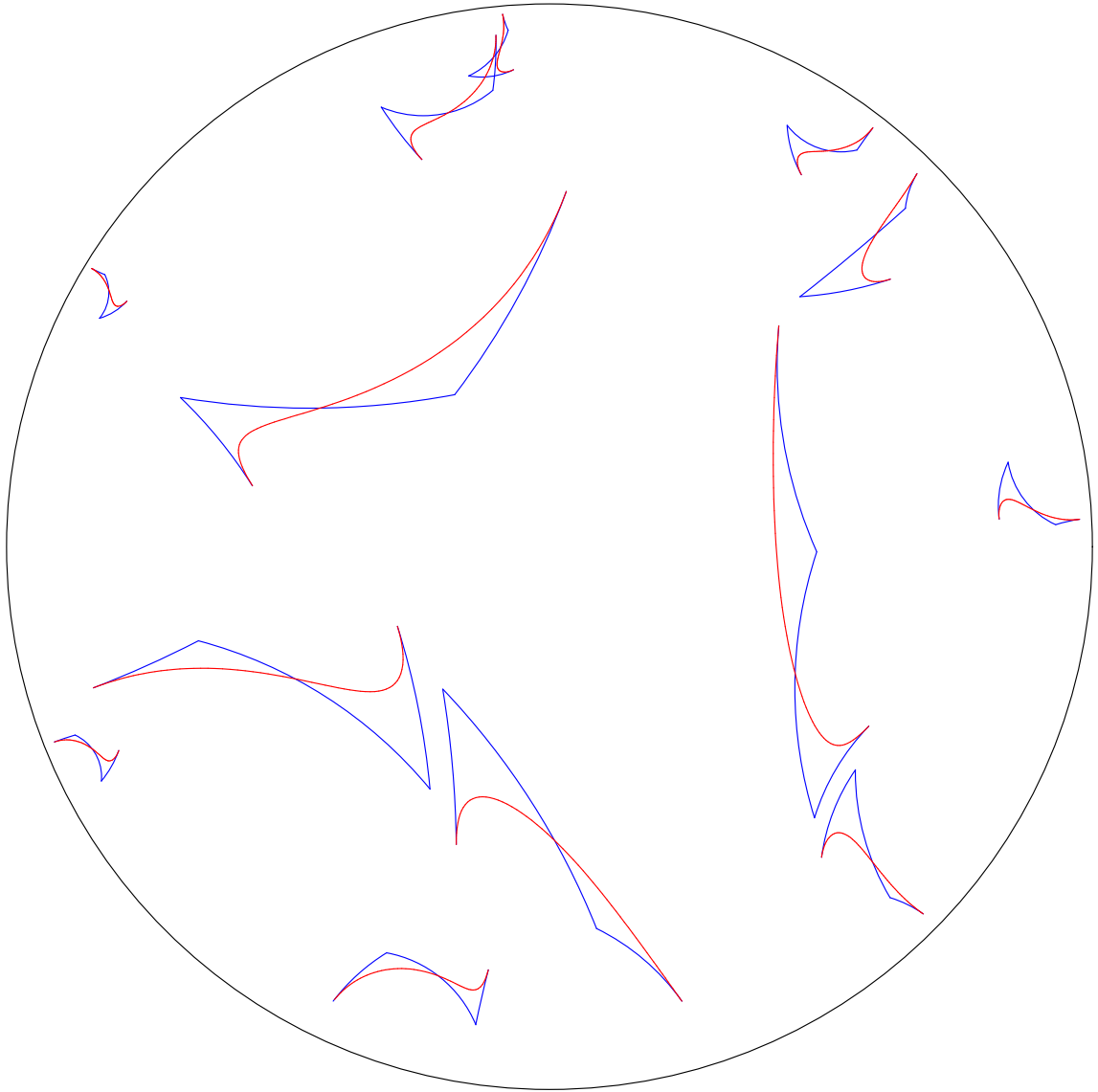


Figure 20: Hyperbolic Spline (red) and Hyperbolic Polyline (blue)

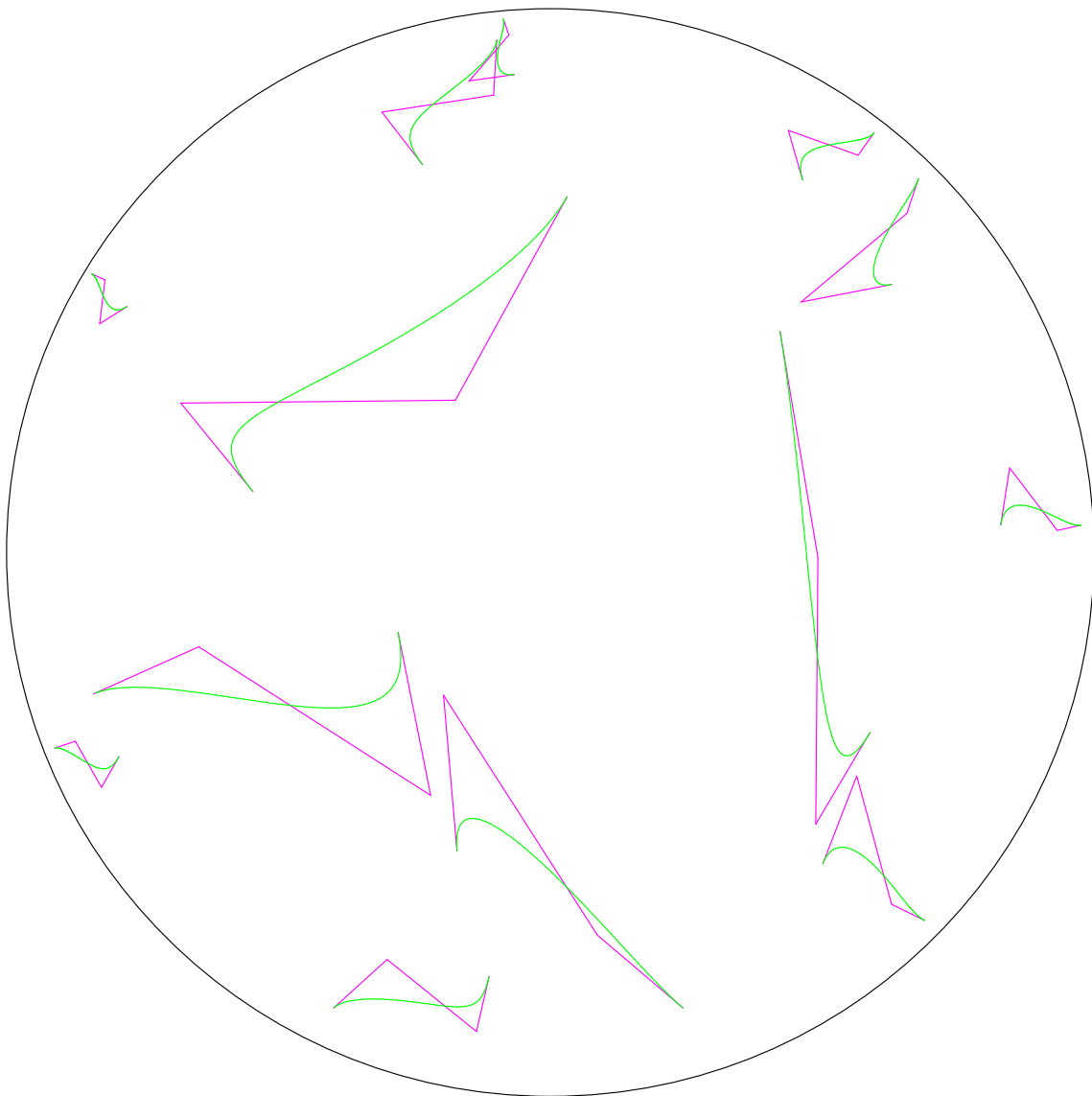


Figure 21: Euclidean Spline (green) and Euclidean Polyline (magenta)

5.2 Hyperbolic and Euclidean Splines

Figures 22 to 31 give five sets of comparisons between hyperbolic splines (red) and Euclidean splines (green).

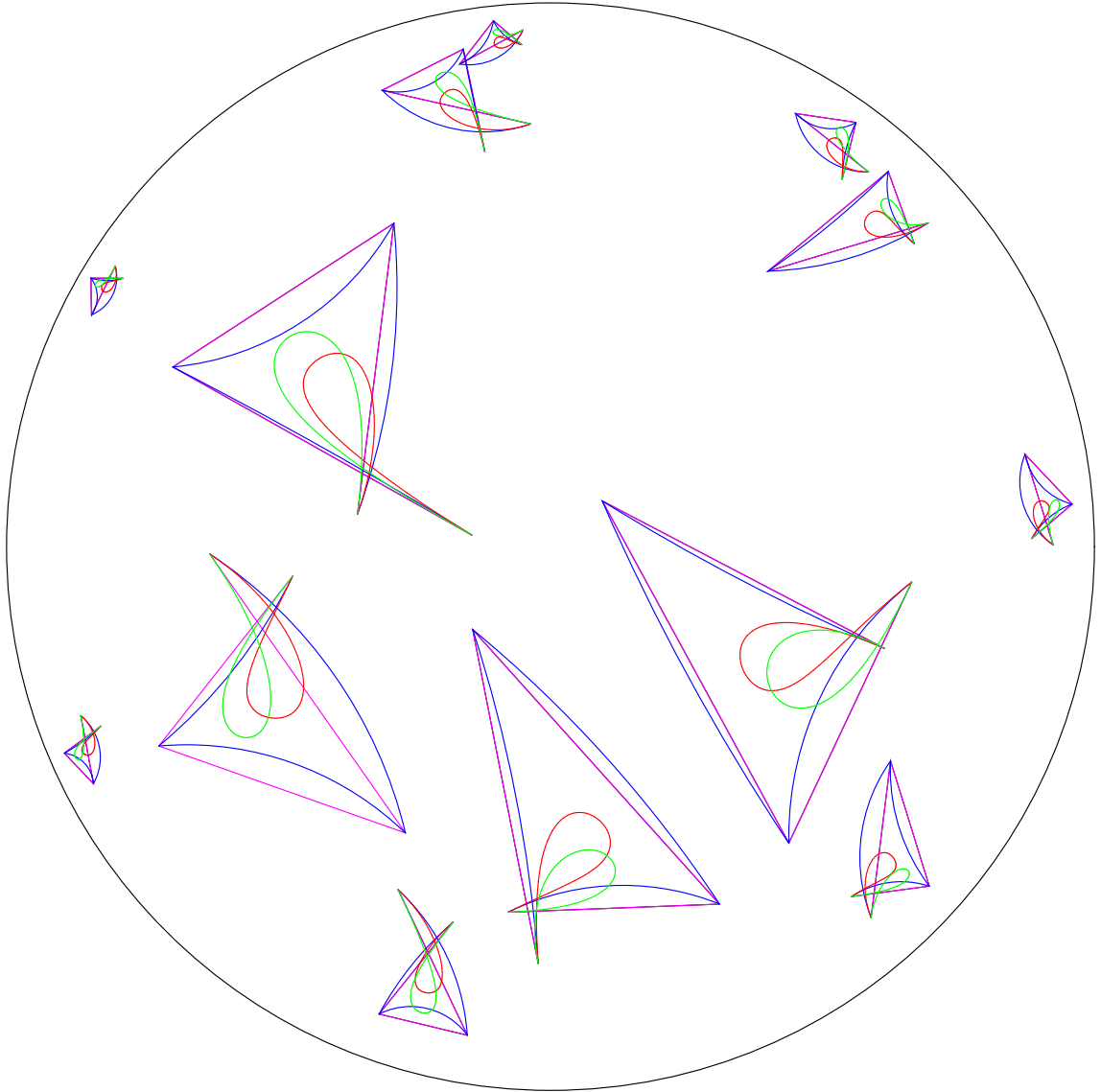


Figure 22: Spline 6

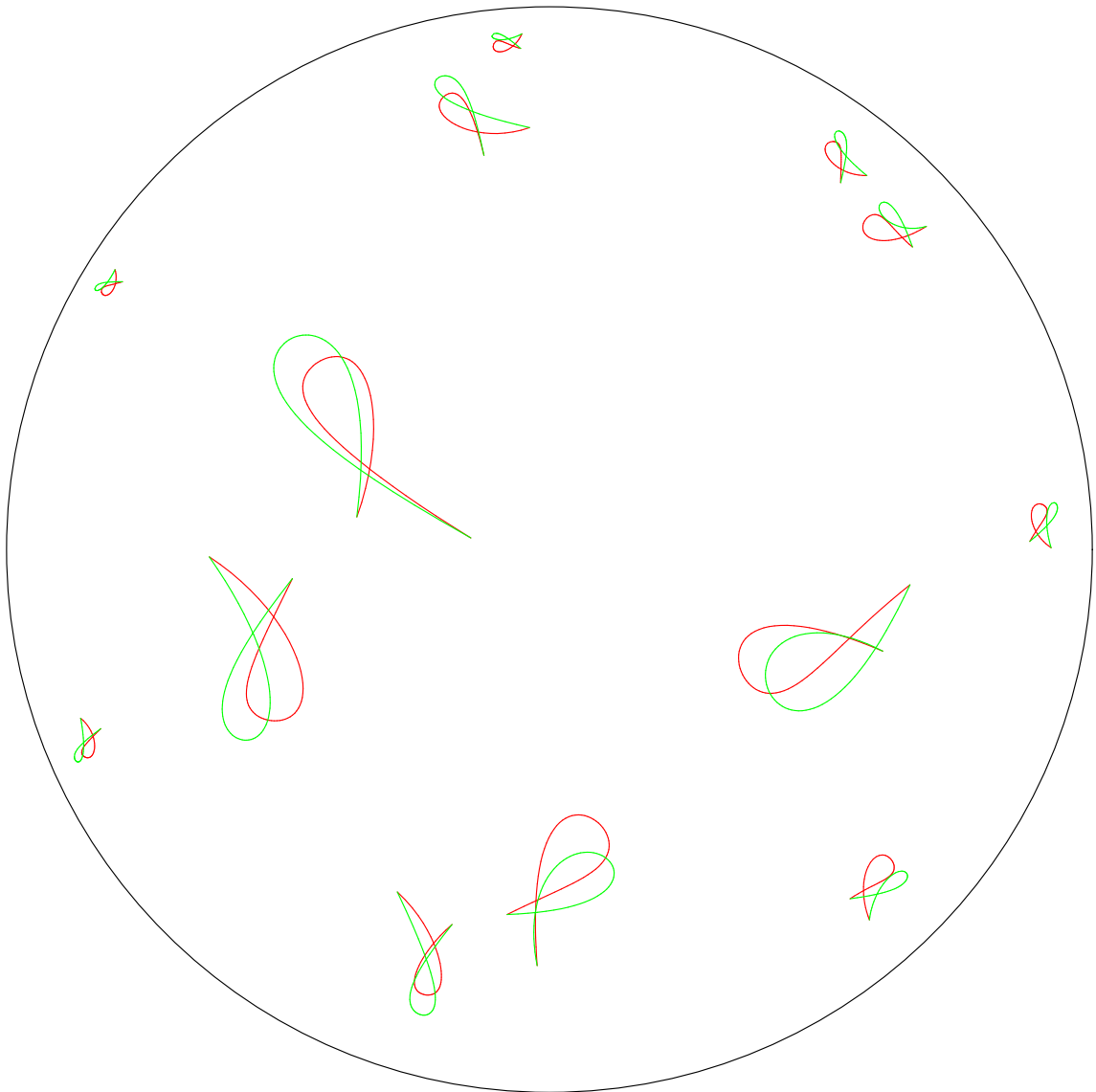


Figure 23: Hyperbolic Spline (red) and Euclidean Spline (green)

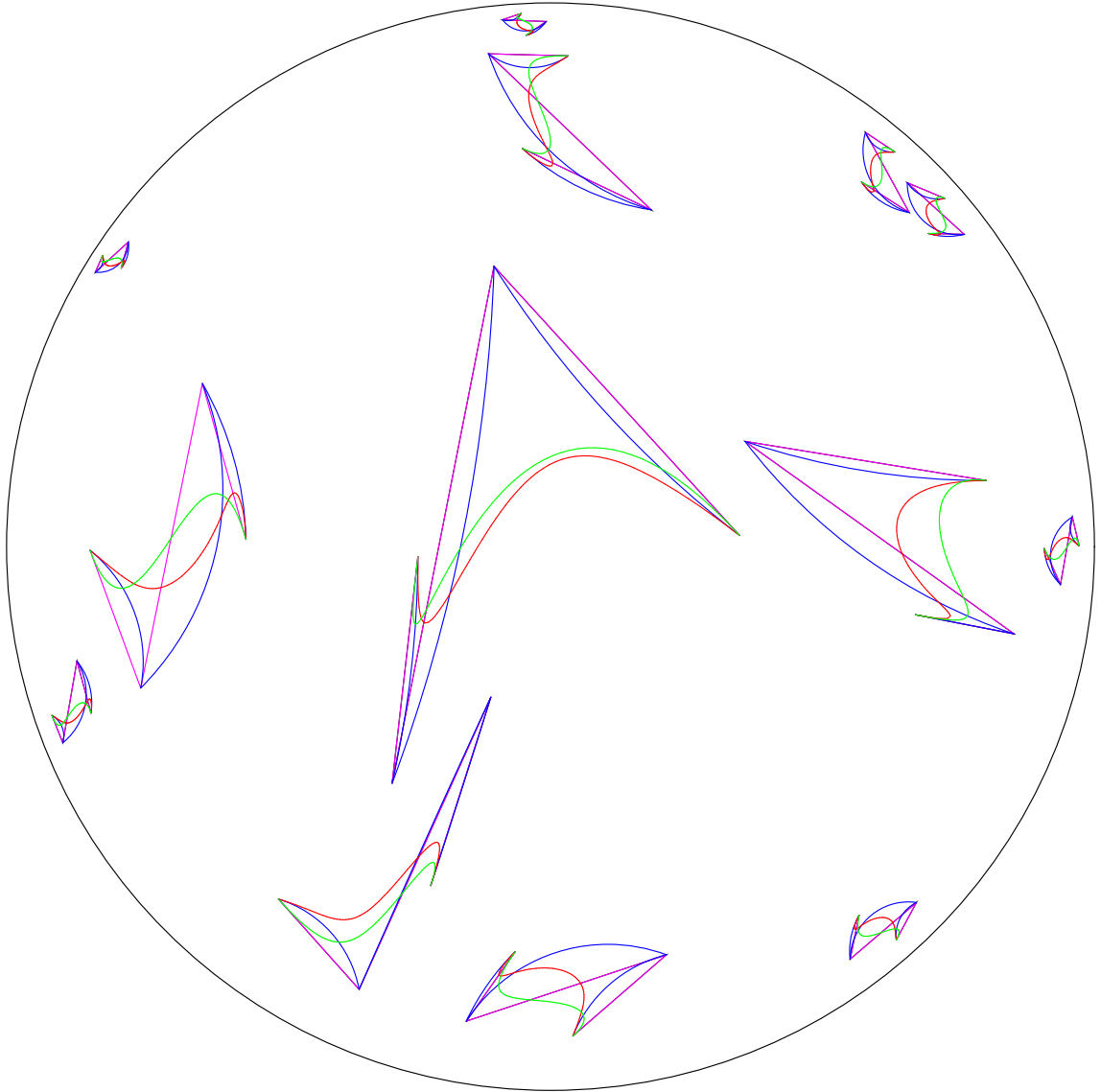


Figure 24: Spline 7

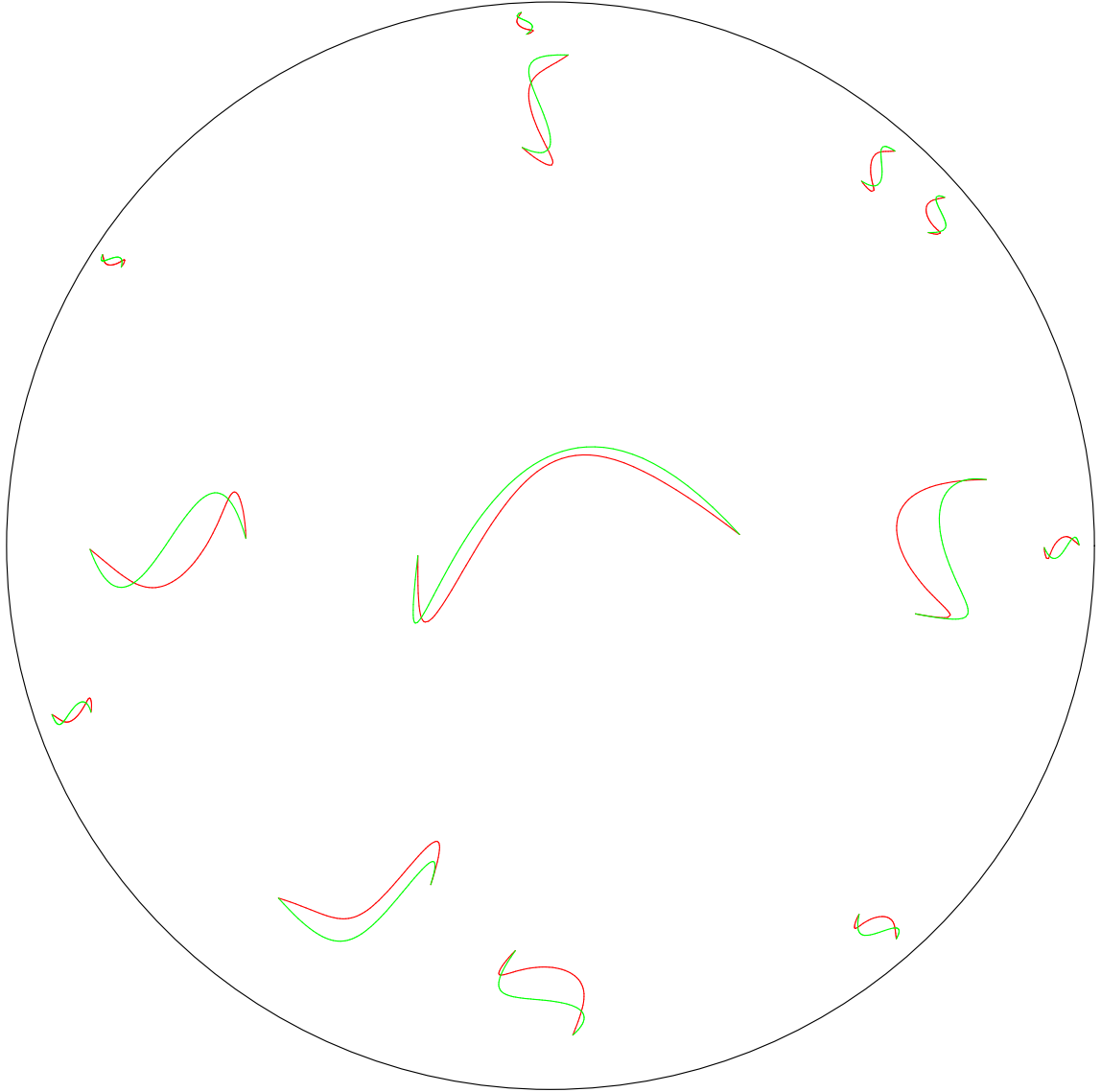


Figure 25: Hyperbolic Spline (red) and Euclidean Spline (green)

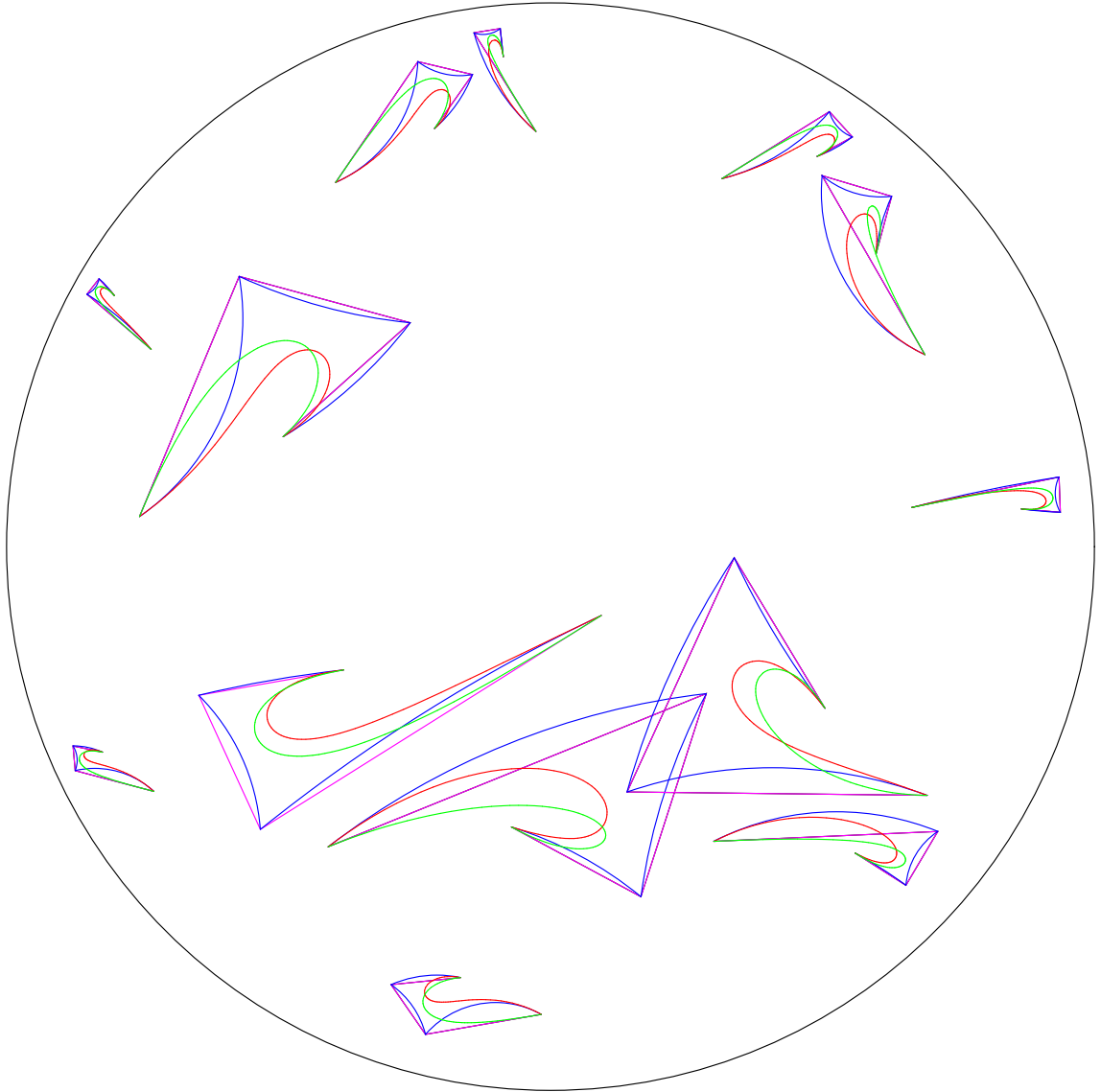


Figure 26: Spline 8



Figure 27: Hyperbolic Spline (red) and Euclidean Spline (green)

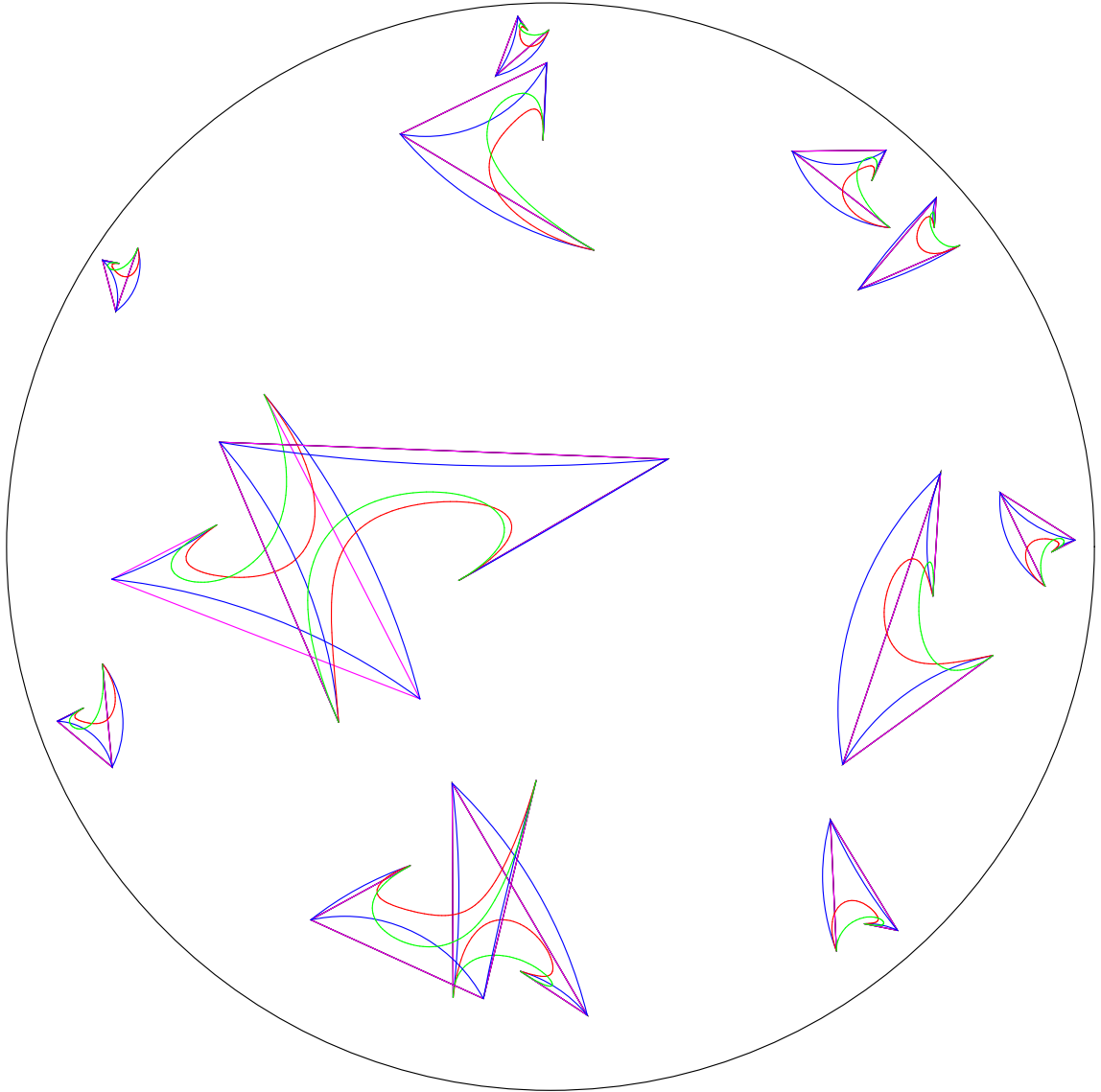


Figure 28: Spline 9

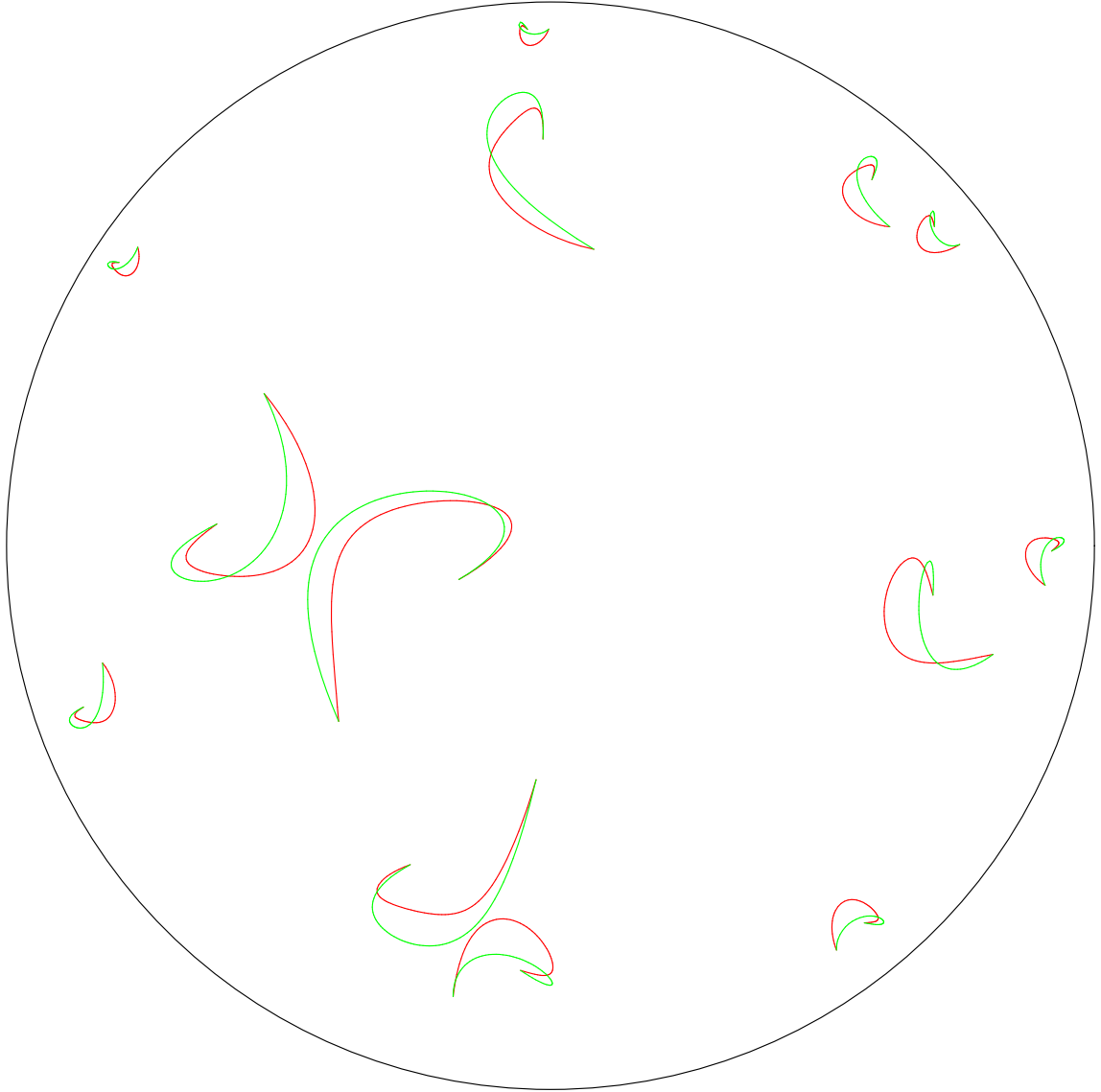


Figure 29: Hyperbolic Spline (red) and Euclidean Spline (green)

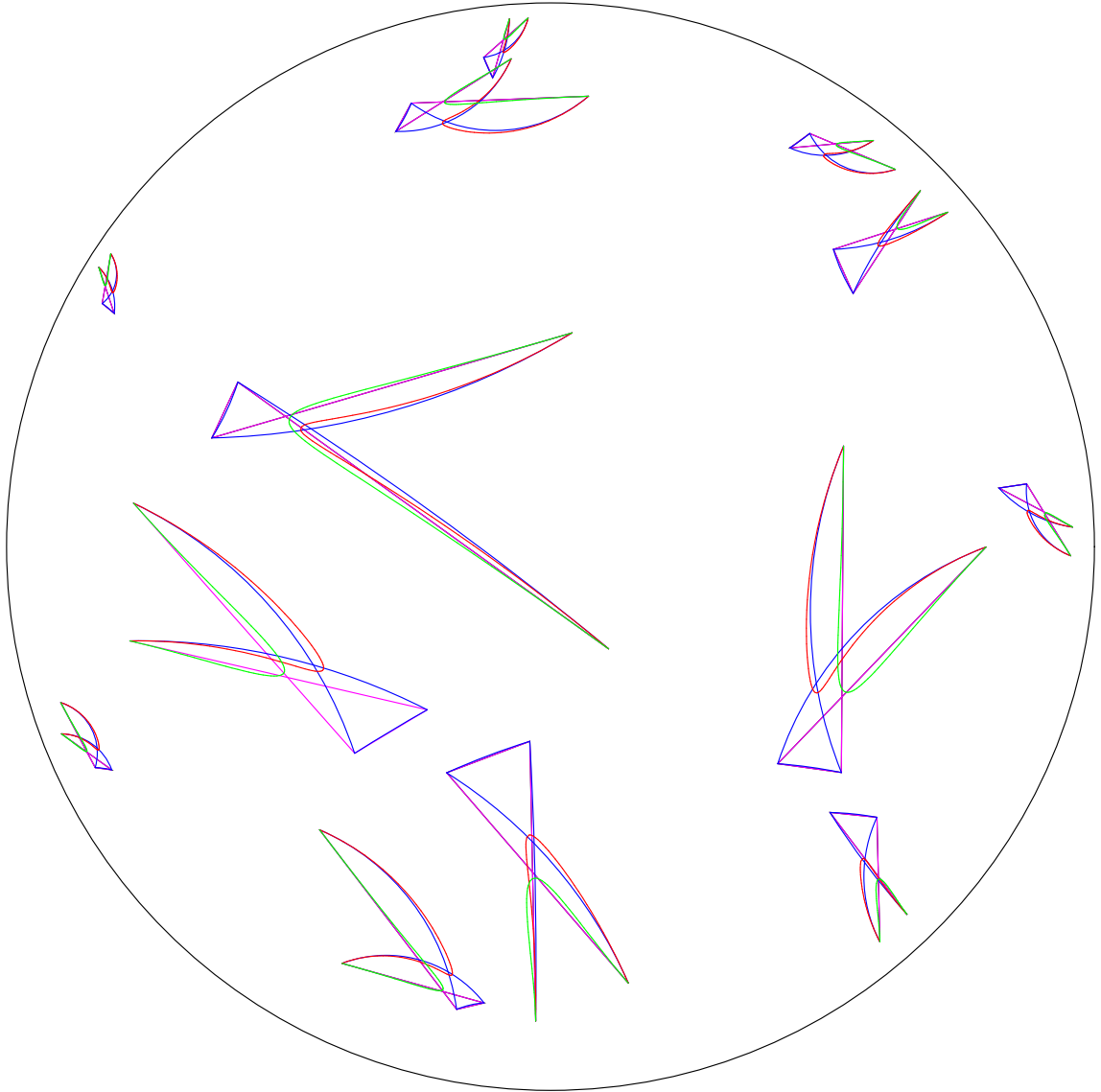


Figure 30: Spline 10

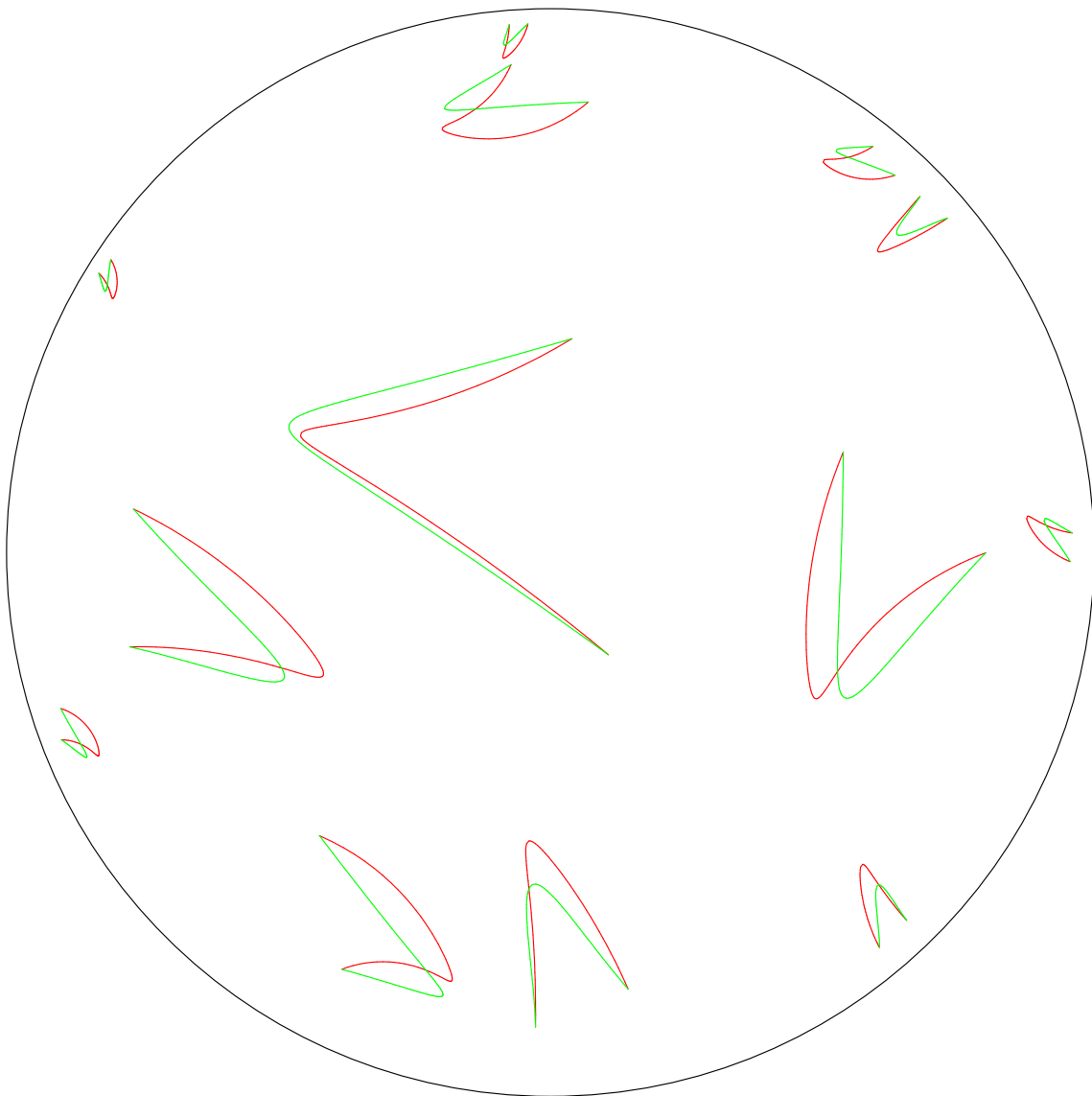


Figure 31: Hyperbolic Spline (red) and Euclidean Spline (green)

5.3 Observation

One observation is that if the original hyperbolic spline drawn is close to the origin (the center of the Poincaré disk in graphs), it is very close to the corresponding Euclidean spline. Meanwhile, the corresponding hyperbolic and Euclidean polylines are very close to each other too. That is the same as we expected, since the closer to the origin, the more similar hyperbolic space is to Euclidean space. We can see some results from Figures 32 to 35.

5.4 Invariance

Hyperbolic distance is invariant under hyperbolic isometries. Therefore we expect that hyperbolic splines will be invariant too, since they are defined in terms of hyperbolic distance.

We designed an experiment to verify it. In the experiment, we still use Professor Dunham's hyperbolic pattern generation program. The first method (call it Method 1) we used to create patterns of hyperbolic splines is that we transfer a subpattern of four control points to get patterns of control points. Then for each four control points, we call `spline.c` to calculate the hyperbolic spline data and draw them. The second method (call it Method 2) is that we only call `spline.c` once to draw the first spline. We then transfer the data for that spline to get repeated patterns of splines.

We programmed both methods and did an experiment. Figure 36 is the result of using Method 1, and Figure 37 is the result of using Method 2. From the results, we can see that both methods produce exactly the same hyperbolic splines. The data for the splines are the same at least for the first four decimal places. That is exactly what we expected since distance in hyperbolic space is invariant. This experiment confirmed the invariance of hyperbolic splines under conformal self-maps.

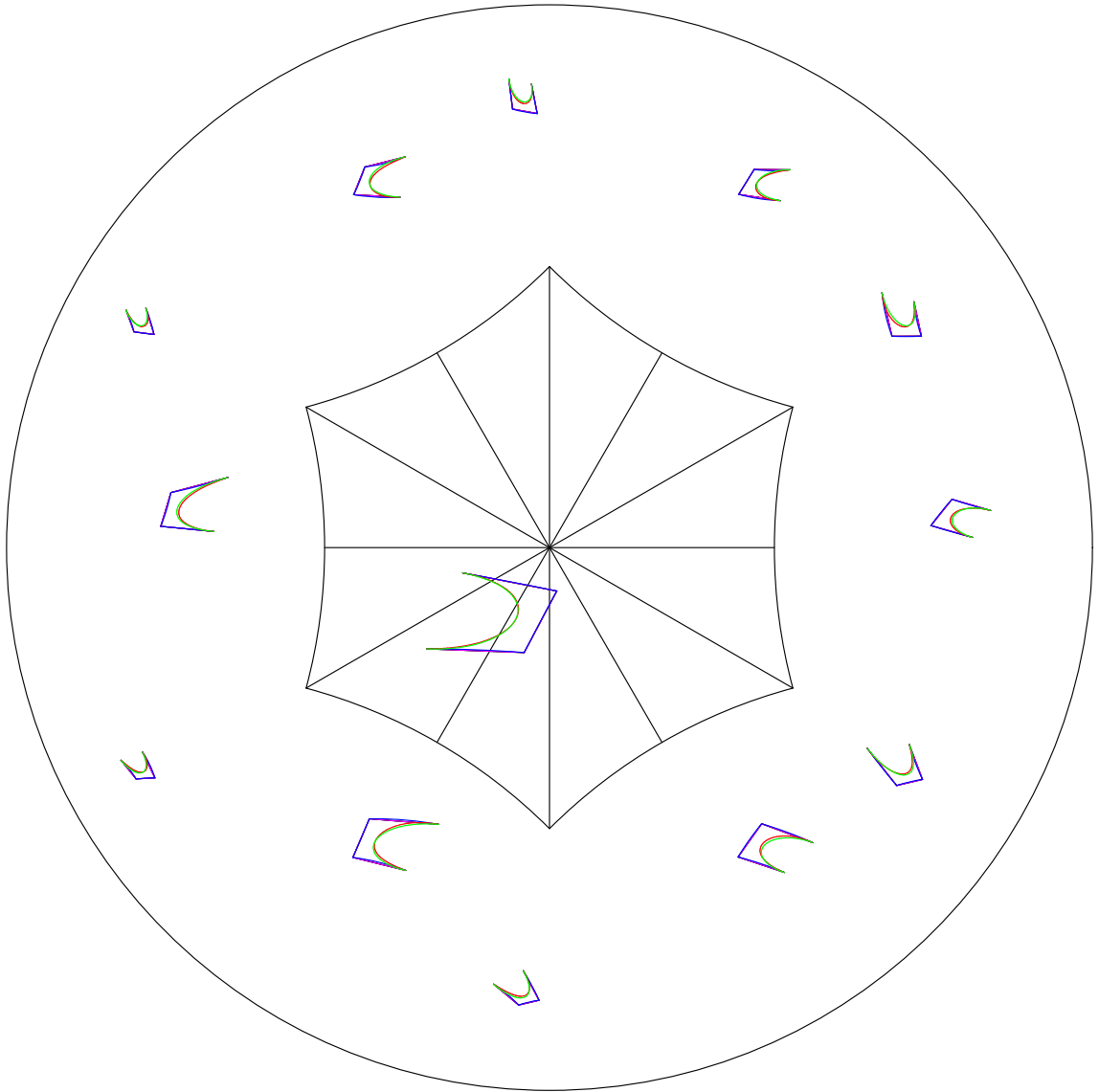


Figure 32: Spline 11

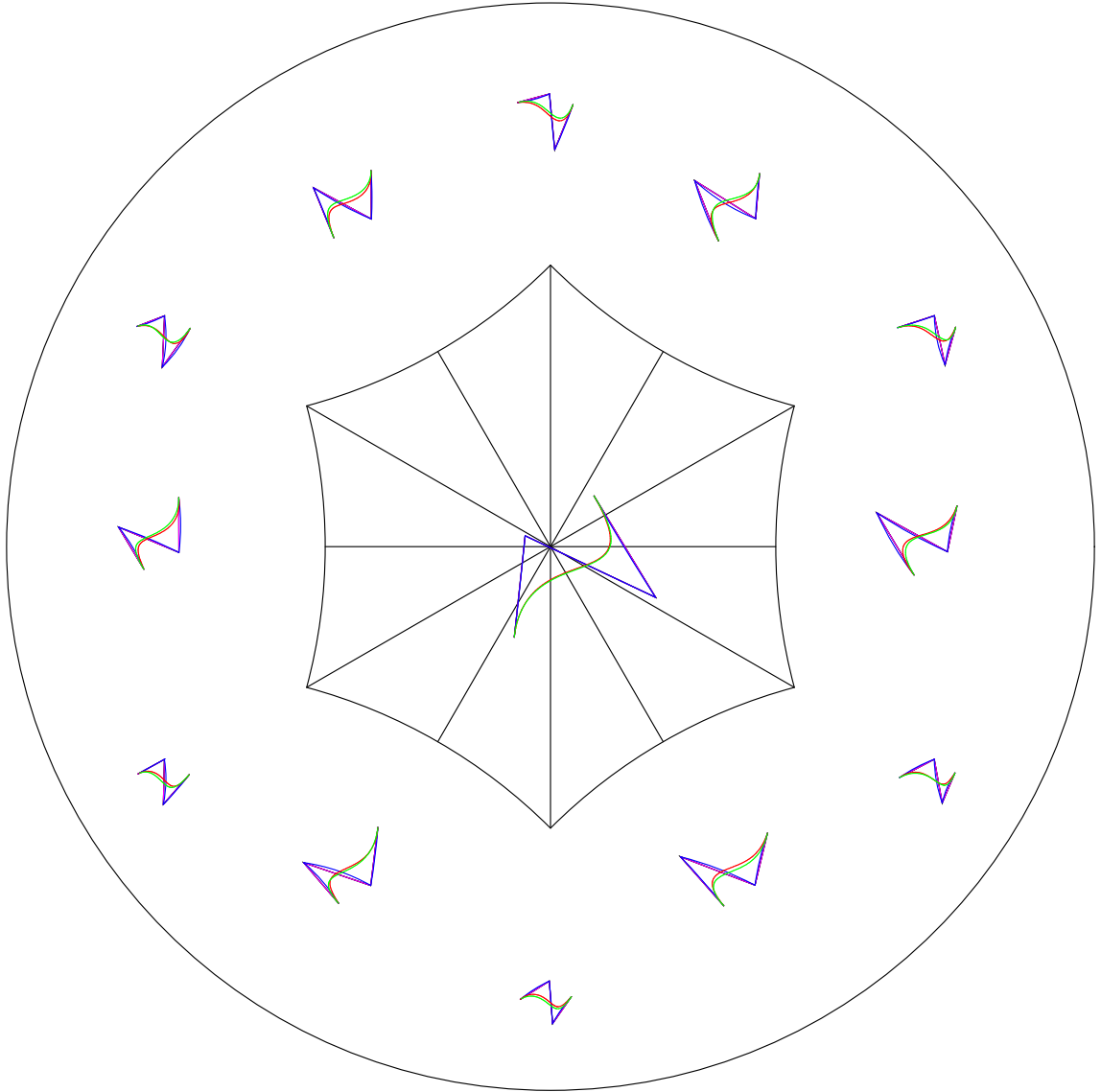


Figure 33: Spline 12

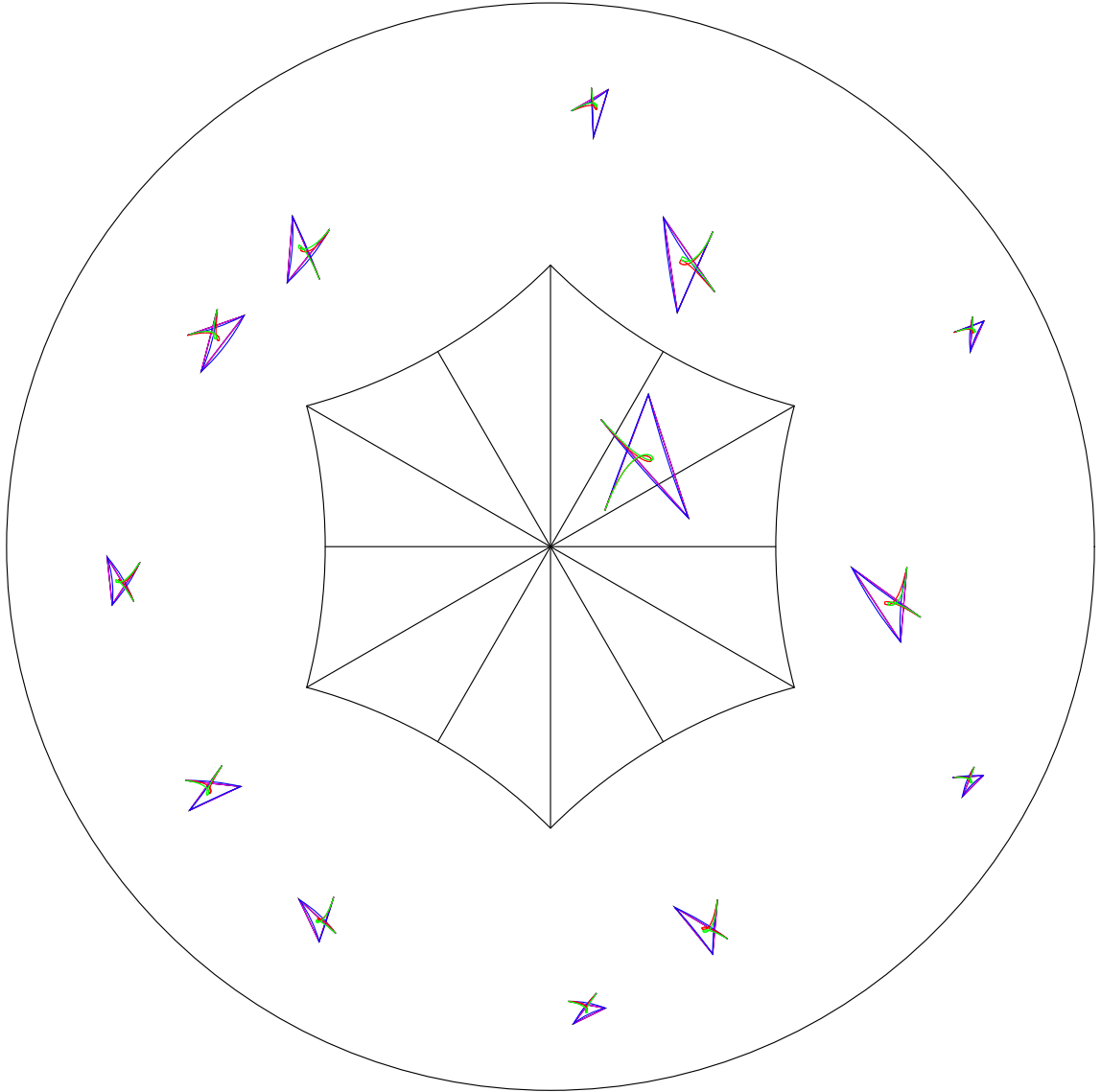


Figure 34: Spline 13

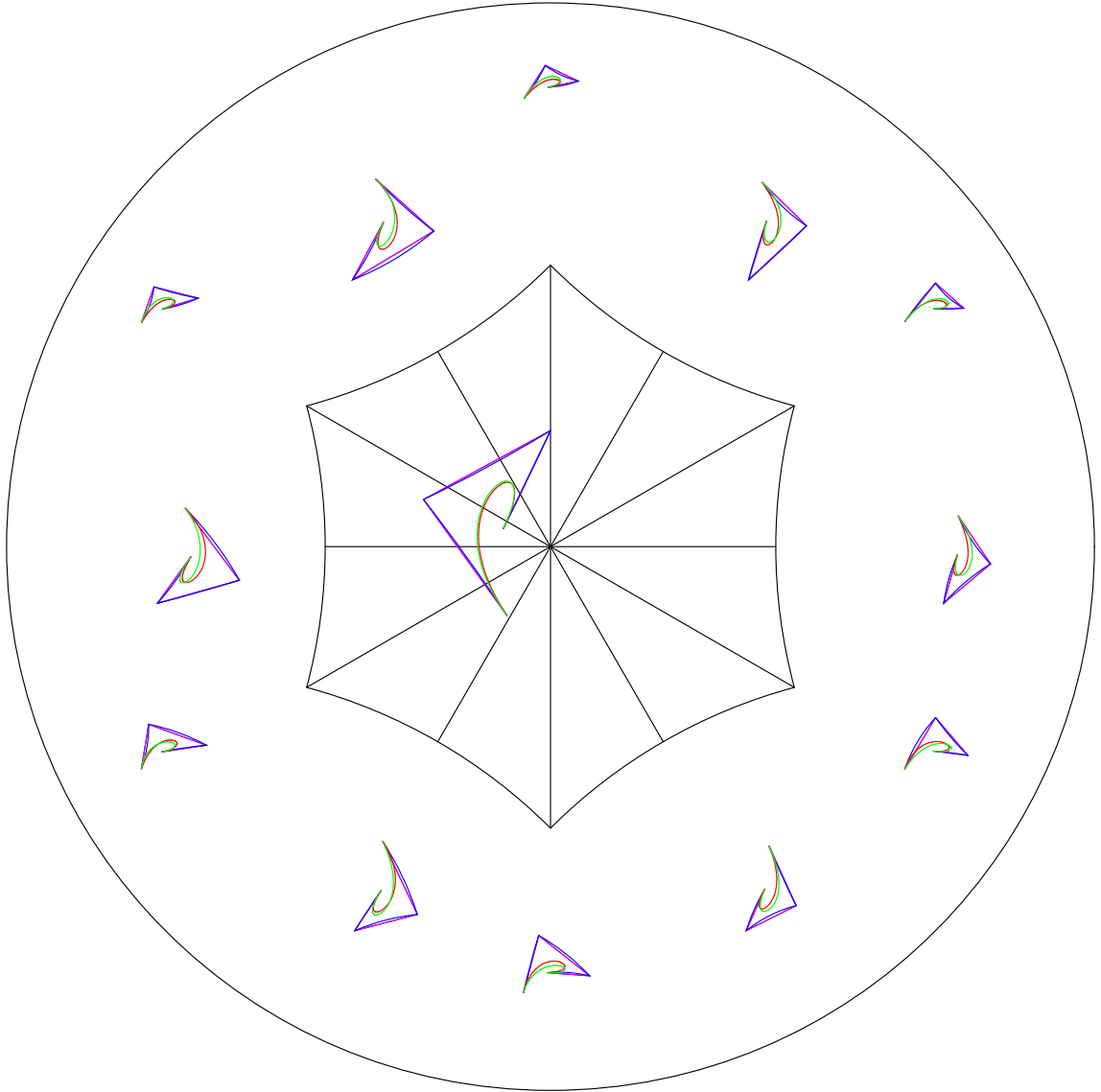


Figure 35: Spline 14

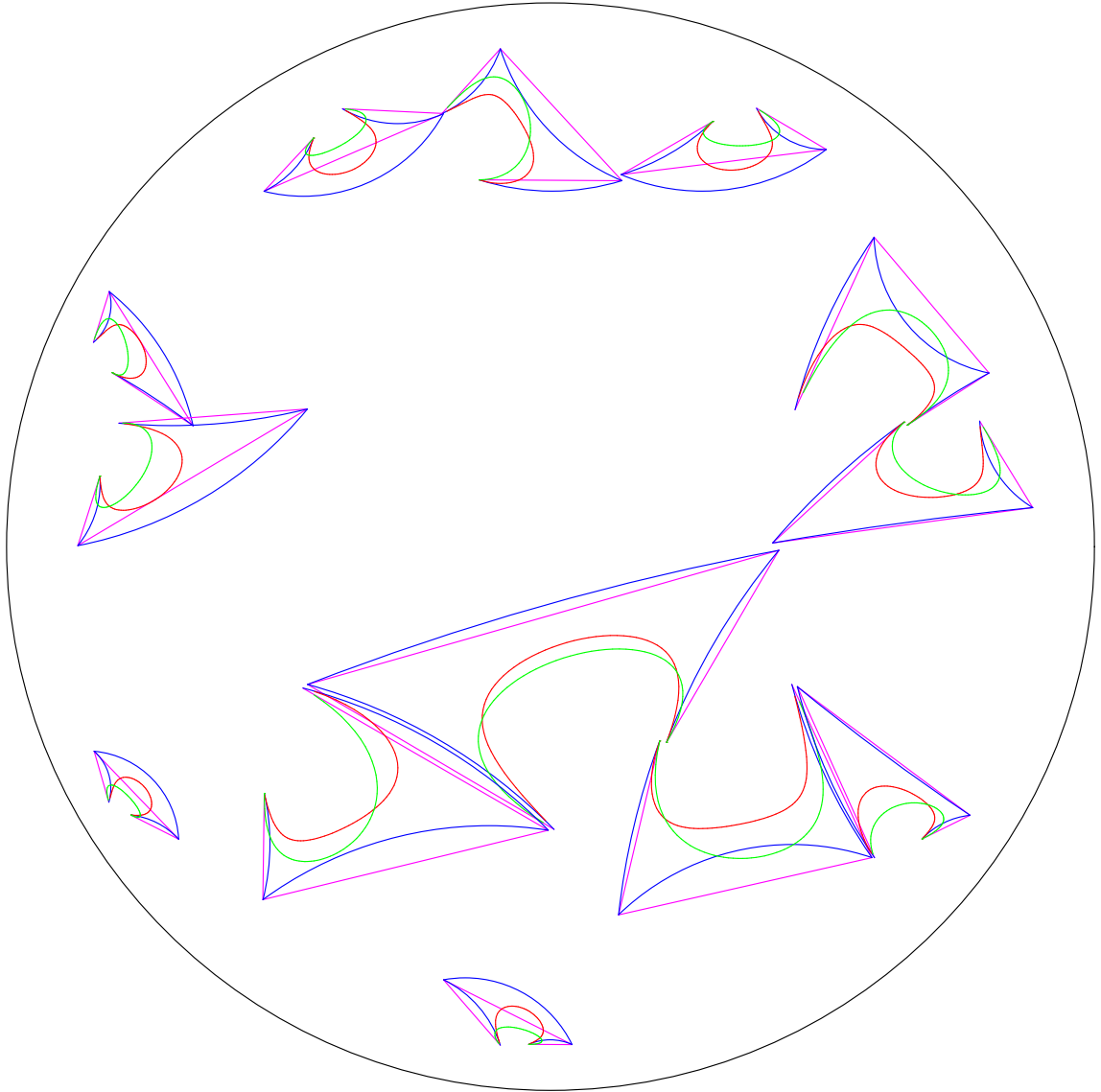


Figure 36: Spline 15 using Method 1

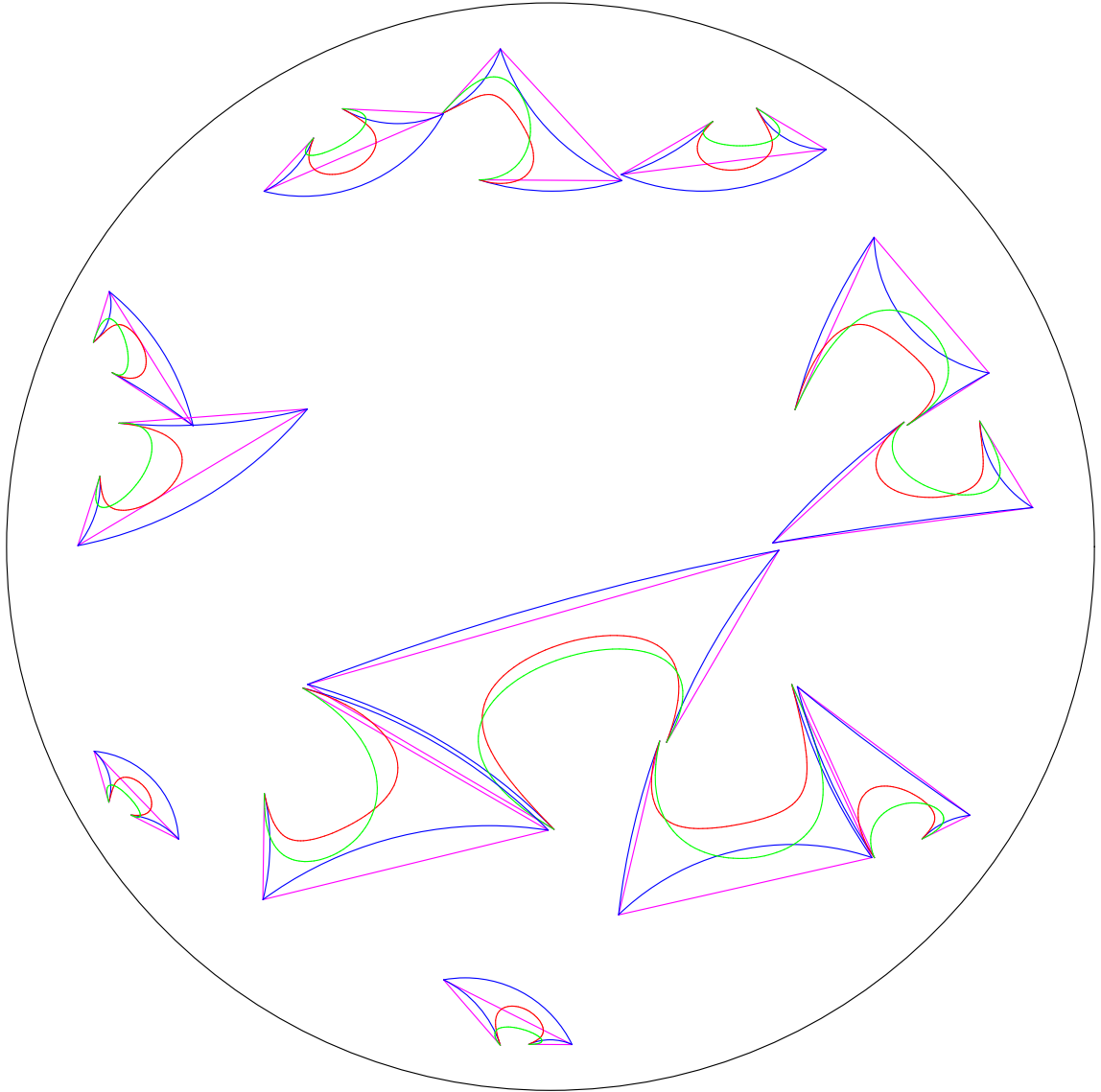


Figure 37: Spline 16 using Method 2

5.5 An Application with the Hyperbolic Fish Pattern

This section shows the result of applying our algorithm to Professor Dunham's hyperbolic fish pattern. Figure 38 is Professor Dunham's original hyperbolic fish pattern. From the graph, we can see that the border line between the fish body and its tail is composed of many small straight lines. Figure 39 is a revised fish pattern. The border line between the fish body and its tail is a polyline with four control points. Applying our algorithm, the program draws a hyperbolic spline with the same four control points as in Figure 39. We can see the result in Figure 40. From the comparison of the three graphs, the improvement by hyperbolic splines in this case is quite noticeable. Thus, we can apply this new algorithm to fish body lines, make the shape nice and smooth and therefore the whole pattern look better.

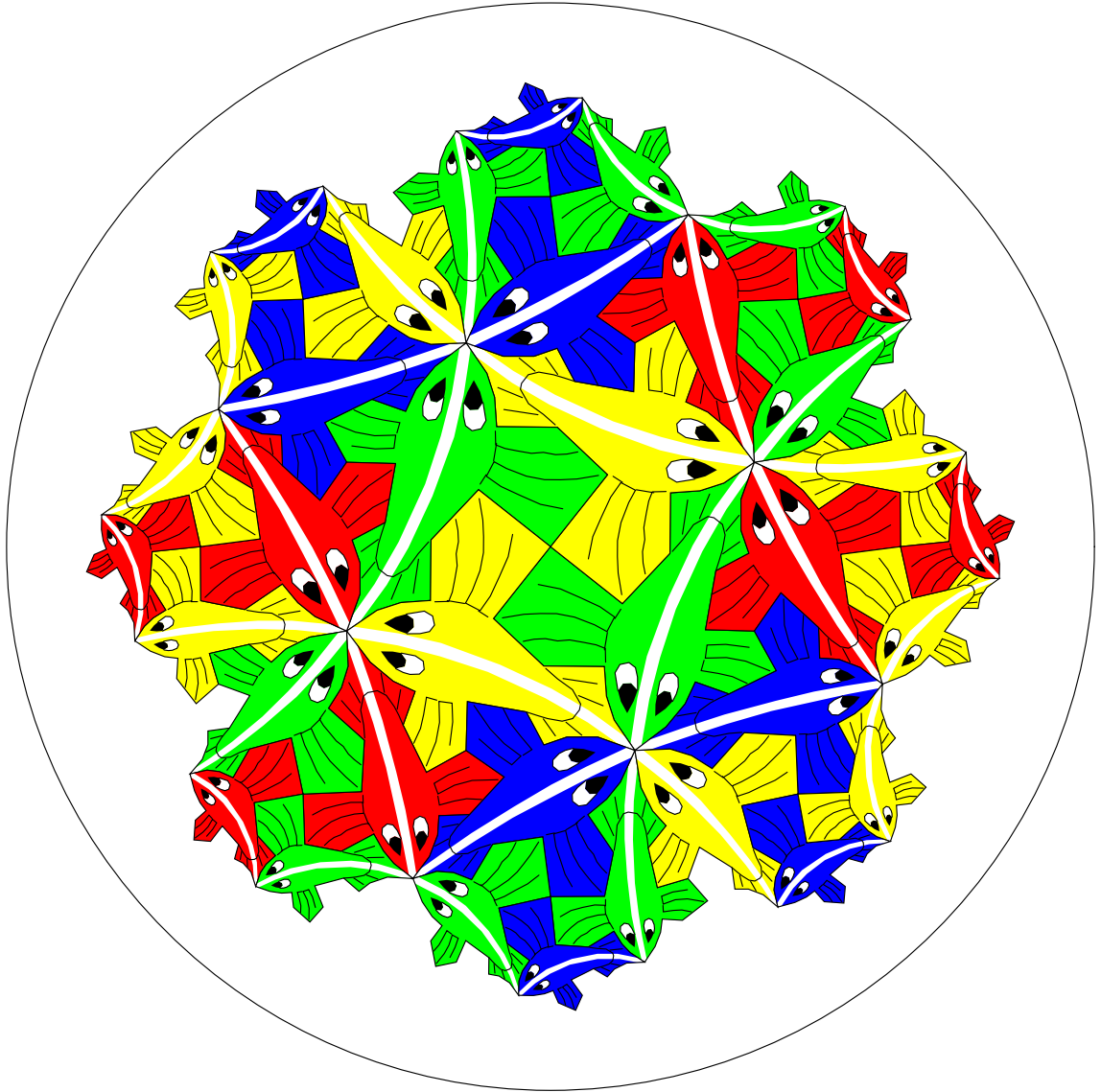


Figure 38: Original Fish Pattern

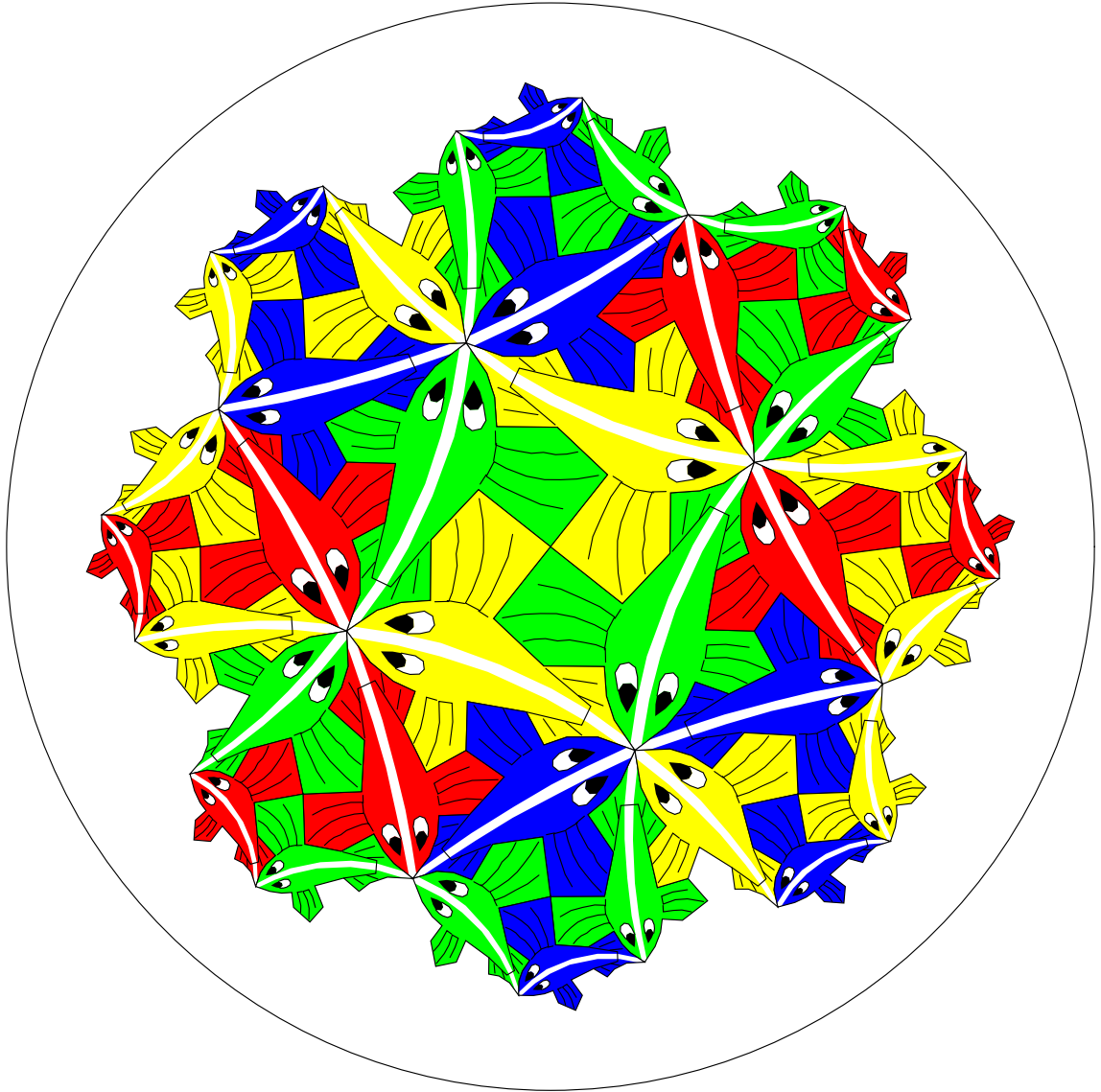


Figure 39: Original Fish Pattern with Four Control Points

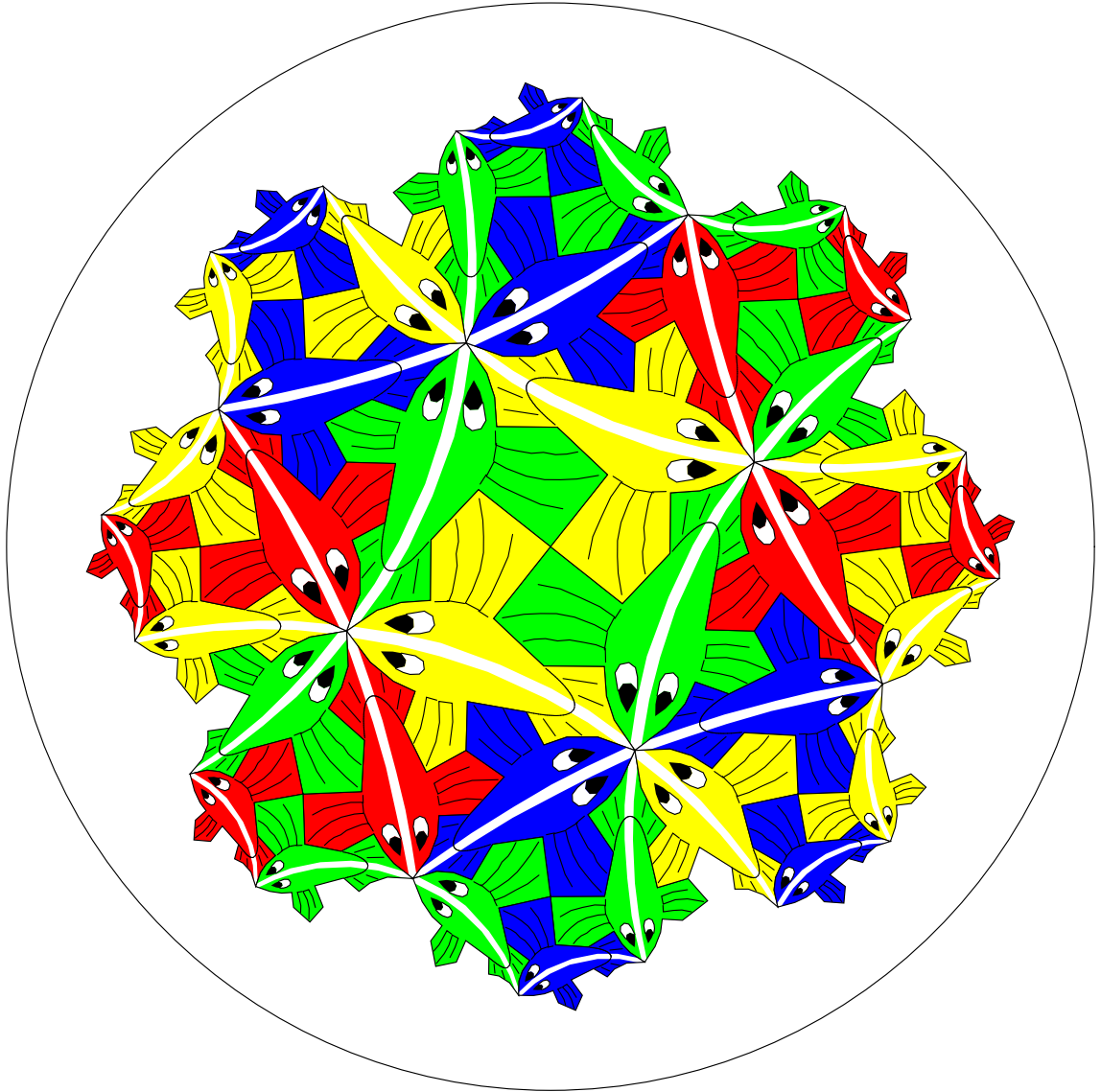


Figure 40: New Fish Pattern with Splines

6 Possible Future Work in this Area

In Buss and Fillmore's paper [1], they provided two different algorithms to calculate a spherical weighted average. One is a linear convergence rate algorithm, the other is a quadratic convergence rate algorithm which uses Newton descent to find a critical point of the function f . The research here only investigated the linear convergence rate algorithm which is an analogue of that for spherical splines. However, the quadratic convergence rate algorithm for hyperbolic splines was not investigated. In [1], Buss and Fillmore experimented with both algorithms and concluded that the quadratic convergence rate algorithm is substantially faster than the linear convergence rate algorithm in lower-dimensional spheres. We are thinking that the quadratic convergence rate algorithm could also be a good algorithm for hyperbolic splines and may provide better computation time. Therefore we could do some future work in this area.

Another area of possible future work lies in the form for hyperbolic splines. The research here uses Bézier curves as the representations for hyperbolic splines, which is

$$Q(t) = w_1P_1 + w_2P_2 + w_3P_3 + w_4P_4,$$

where the weights are $w_1 = (1 - t)^3$, $w_2 = 3t(1 - t)^2$, $w_3 = 3t^2(1 - t)$, $w_4 = t^3$.

We could investigate some other weighting functions such as those for Hermite curves and B-splines [2].

7 Conclusion

Since weighted average splines have been defined in Euclidean and spherical space, this research fills a gap by defining such splines in hyperbolic space, the third of the “classical” geometries.

An immediate contribution is that it provides Professor Dunham with accurate hyperbolic spline curves for his hyperbolic pattern generating program. More generally, this research might lead to an understanding of weighted average spline curves in Riemannian manifolds.

References

- [1] Buss, Samuel R., Jay Fillmore, *Spherical Averages and Applications to Spherical Splines and Interpolation*, ACM Transactions on Graphics, Vol. 20, No. 2, April 2001, Pages 95-126.

- [2] Foley, James, A. Van Dam, S. Feiner, and J. Hughes, *Computer Graphics: Principles and Practice*, Addison Wesley, 1990, 1996.

- [3] Lath, Amit, *Approximate Hyperbolic Splines and Their Transformations*, Master of Science thesis, Department of Computer Science, University of Minnesota Duluth, 2002.

- [4] de Boor, Carl, an on-line spline bibliography at <http://www.cs.wisc.edu/~deboor/bib/bib.html>
an up-to-date (as of July, 2004) searchable on-line bibliography of spline-related books and papers.

- [5] Faber, Richard L., *Foundations of Euclidean and Non-Euclidean Geometry*, Marsel Dekker, Inc., New York and Basel, 1983.

- [6] Dunham, Douglas, *Hyperbolic Symmetry*, Comp. & Maths. with Appls., Vol. 12B, Nos. 1/2, pp 139-153, 1986.