The Cross Ratio and Quadrilaterals

Sept. 7, 2016

This is an exploration of what the cross ratio means for quadrilaterals. Just to make things a little easier, we will specialize to the case of planar quadrilaterals.

Recall, we're interested in equilateral polygons up to translation and rotation, so we can represent our polygons by an ordered tuple of unit vectors – or points on the circle – up to rotation.

Now, if we just choose a random n-tuple of points $(\vec{p}_1, \ ..., \ \vec{p}_n)$ on the circle, we will get a random walk in the plane, but with probability 1 it will not close up to form a closed polygon.

What does it mean to be a closed polygon? Well, it means the vector $\vec{p}_1 + ... + \vec{p}_n = \vec{0}$ or, equivalently, that the **center of mass** of the $(\vec{p}_1, \ ..., \ \vec{p}_n)$ is at the origin. If there were a unique way to associate a **centered** point cloud $(\vec{e}_1, \ ..., \ \vec{e}_n)$ with an arbitrary point cloud $(\vec{p}_1, \ ..., \ \vec{p}_n)$, then we would get a map from random walks to random polygons.

This doesn't quite work (things can go wrong if too many of the $\vec{p}_i$ are equal), but if we restrict to the case where all the $\vec{p}_i$ are **distinct**, then it turns out that there is as way to do this.

The key is the simple but deep observation that Möbius transformations of the circle extend to maps of the unit disk bounded by the circle, and that these maps are exactly the group of hyperbolic isometries in the Poincaré disk model of the hyperbolic plane. Now, using this connection to hyperbolic geometry, Douady and Earle (http://dx.doi.org/10.1007/BF02392590) showed that, up to rotations, there is a unique Möbius transformation sending a point cloud $(\vec{p}_1, \ ..., \ \vec{p}_n)$ to a new point cloud $(\vec{e}_1, \ ..., \ \vec{e}_n)$ whose center of mass is at the origin. It's clear why this is only "up to rotations", since any rotation of a point cloud whose center of mass will be at the origin will still have center of mass at the origin.

Since we only care about our polygons up to rotation anyway, the fact that we only have uniqueness up to rotation isn't a concern.

Another way of stating Douady and Earle's result is the following: each point in $M_{0,n}(\mathbb{R})$ has a representative with center of mass at the origin, and this representative is unique up to rotation. So we can think of $M_{0,n}(\mathbb{R})$ as (almost) the space of equilateral planar n-gons up to translation and rotation (the "almost" is because there exist polygons with parallel edges which cannot come from points in $M_{0,n}(\mathbb{R})$, but such polygons obviously form a closed subset of positive codimension, and so are irrelevant from a measure theory perspective).

Now, let's specialize to quadrilaterals, so $n = 4$.

By the same argument as in the complex case, we get a global coordinate on $M_{0,4}(\mathbb{R}) \simeq \mathbb{RP}^1 \setminus \{0, 1, \infty\}$ given by $t \mapsto (0, 1, \infty, t)$ with inverse map given by $(p_1, p_2, p_3, p_4) \mapsto \mathrm{CR}(p_1, p_2, p_3, p_4)$. The question, very simply, is: what does the cross ratio tell us about the shape of a quadrilateral?

Here's the strategy: define the map from $\mathbb{RP}^1 \setminus \{0, 1, \infty\} \simeq \mathbb{R} \setminus \{0, 1\}$ to $M_{0,4}(\mathbb{R})$, find the unique (up to rotation) representative which corresponds to a closed quadrilateral, then display it and see what happens as we vary $t \in \mathbb{R} \setminus \{0, 1\}$.

Infrastructure

First, we need to write all the code to implement the above steps.

Remember, we want to input a bunch of points on the unit disk and then find a Möbius transformation that sends them to points whose center of mass is at the origin.

Since we could always post-compose with a rotation and get another collection of points with center of mass at the origin, we can restrict to those Möbius transformations which are "orthogonal" (in some sense) to the rotations. It turns out to be enough to be able to translate any point $\mu \in \mathbb{D}$ to the origin along the hyperbolic geodesic connecting μ to the origin. This Möbius transformation $\phi_\mu(z)$ can be written as:

```
HyperbolicIsometry[z_, μ_] := (z - μ)/(1 - Conjugate[μ] z);
```

Now, the only tricky bit of this whole story is figuring out which of the ϕ_μ defined above will map a given point cloud on the circle to a centered point cloud. Douady and Earle's method involved Busemann functions and conformal barycenters and a bunch of other nasty and impossible-to-compute gadgets, but there's a computational much more tractable way apparently due to Milnor, though communicated by Abikoff–Ye (http://dx.doi.org/10.1090/conm/211/02811 and http://dx.doi.org/10.1007/BF02786649)

The idea is actually very simple. If c_1 is the (Euclidean) center of mass of the starting point cloud $(\vec{p}_1, \ldots, \vec{p}_n)$, then ϕ_{c_1} sends c_1 to the origin; apply ϕ_{c_1} to each of the $\vec{p}_i$ to get a new point cloud $(\vec{p}_{2,1}, \ldots, \vec{p}_{2,n}) = (\phi_{c_1}(\vec{p}_1), \ldots, \phi_{c_1}(\vec{p}_n))$. It is very unlikely that $(\vec{p}_{2,1}, \ldots, \vec{p}_{2,n})$ has center of mass c_2 at the origin, but the key result is that c_2 is closer to the origin (in the hyperbolic metric) than c_1 was. But now iterate: apply ϕ_{c_2} to $(\vec{p}_{2,1}, \ldots, \vec{p}_{2,n})$ to get $(\vec{p}_{3,1}, \ldots, \vec{p}_{3,n}) = (\phi_{c_2}(\vec{p}_{2,1}), \ldots, \phi_{c_2}(\vec{p}_{2,n})) = (\phi_{c_2}(\phi_{c_1}(\vec{p}_1)), \ldots, \phi_{c_2}(\phi_{c_1}(\vec{p}_n)))$, etc. Then the limit as $k \to \infty$ of

$$(\vec{p}_{k,1}, \ldots, \vec{p}_{k,n}) = (\phi_{c_k} \circ \ldots \circ \phi_{c_1}(\vec{p}_1), \ldots, \phi_{c_k} \circ \ldots \circ \phi_{c_1}(\vec{p}_n))$$

is exactly the centered point cloud $(\vec{e}_1, \ldots, \vec{e}_n)$ that we want.

So here's an implementation, where we just go until the failure-to-close vector (a.k.a. the sum of the edges, or the distance between the starting point and the ending point of the walk) drops below 10^{-14}.

```
MAYiterationComplex[Z_] := HyperbolicIsometry[#, Mean[Z]] & /@ Z;
MAYiterationAngles[Θ_] := Arg /@ MAYiterationComplex[Exp[I #] & /@ Θ];
FailureToClose[Θ_] := Norm[Total[{Cos[#], Sin[#]} & /@ Θ]];
GITquotient[Θ_] := NestWhile[MAYiterationAngles,
   1.0 * Θ, TrueQ[Norm[FailureToClose[#]] > 10^(-14.0)] &];
```

Some drawing code:

In[318]:=
```
Polygonalize[Θ_] := {Thick, Arrowheads[Large],
   Arrow[ {Re[#], Im[#]} & /@ Join[{0}, Accumulate[Exp[I #] & /@ Θ]]]};
```

Inverse stereographic projection, to move from $\mathbb{R}$ to $\mathbb{RP}^1$. The slightly unusual thing here is that I'm representing points on $\mathbb{RP}^1 = S^1$ as angles, so this is the usual inverse stereographic projection composed with *atan2*.

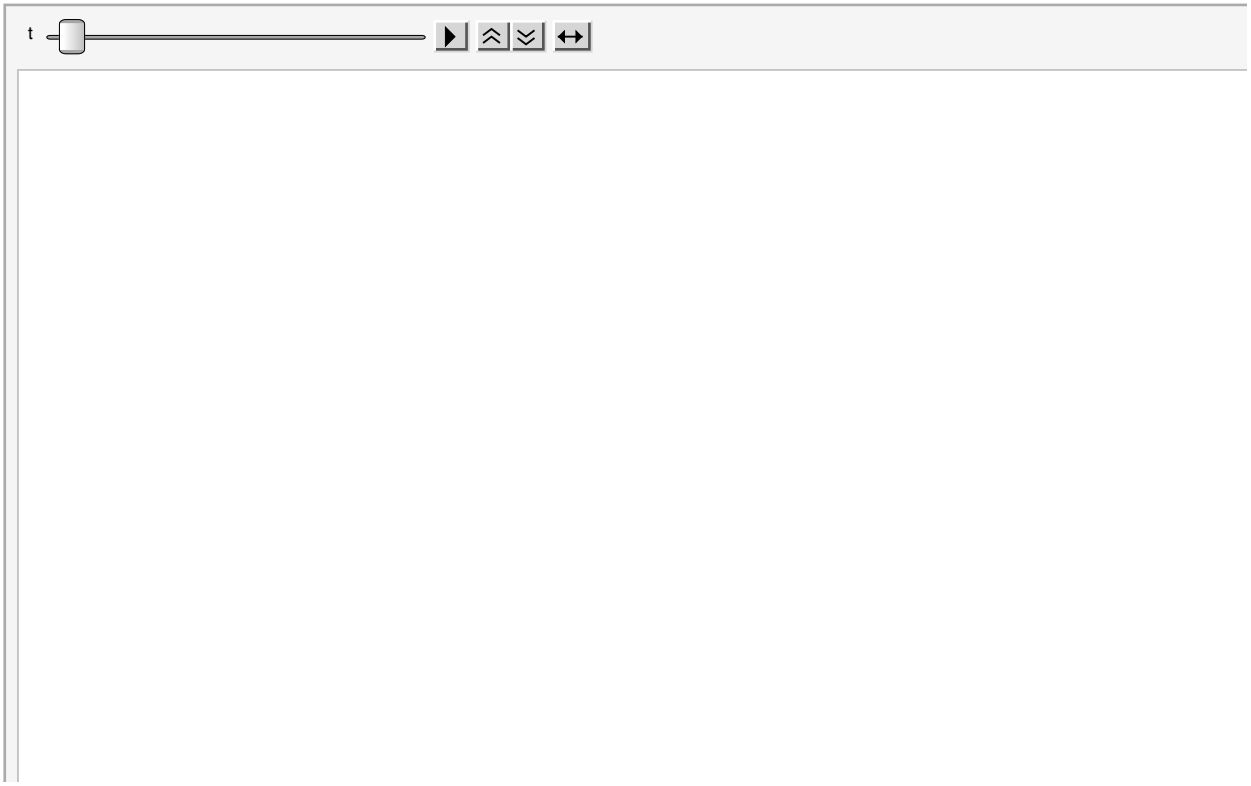
```
InverseStereo[x_] := ArcTan[2 x/(x^2 + 1), (x^2 - 1)/(x^2 + 1)];
```

A global coordinate on quadrilateral space

Now, what are we doing? The idea is that $t \in \mathbb{R} \setminus \{0, 1\}$ is supposed to parametrize quadrilaterals by first mapping into $M_{0,4}(\mathbb{R})$ via $t \mapsto (0, 1, \infty, t)$, and then finding the unique (up to rotation) representative of $[(0, 1, \infty, t)]$ with center of mass at the origin using the Milnor–Abikoff–Ye iteration.

So I build an **Animate** object with the slider giving t, and then perform the above sequence of mappings and draw the resulting quadrilateral. Note that in our angle coordinates on $\mathbb{RP}^1$, the points 0, 1, and ∞ correspond to $-\frac{\pi}{2}$, 0, and $\frac{\pi}{2}$, respectively, so I'm actually applying the M–A–Y iteration to the point $\left(-\frac{\pi}{2}, 0, \frac{\pi}{2}, \textbf{InverseStereo}[t]\right)$. Also, there's a singularity when $t = 0$ or $t = 1$ (or $t = \infty$, though we can't go out that far anyway), so I choose step sizes for the slider to avoid those values.

In[330]:=
```
Animate[
 Graphics[Polygonalize[GITquotient[{-π / 2, 0, π / 2, InverseStereo[t]}]],
  PlotRange → {{-1.1, 1.1}, {-2, 1}},
  PlotLabel → Row[{Style["t", Italic], "=", N[t]}]],
 {t, -10., 10, 20 / 1001}, AnimationRate → 50., AnimationDirection → ForwardBackward]
```



t=−10.

Out[330]=