Reinhardt Project Summary

Matthew Gerstbrein
mlg113@pitt.edu

June 27,2017

This is the summary of the project associated to the Reinhardt Conjecture, conducted in the summer of 2017 (primarily May and June). The intended outcome of the project is to find a local (and ideally global) minimum set of controls corresponding to a hexagonally symmetric, closed shape whose cost is lower than that of the circle. It should be noted that this paper is not self-contained; all terms and concepts referenced in this paper are directly related to my own code, Control Trajectory.ipynb (found in the reinhardt-control-trajectory repository on github under my account, matthew-gerstbrein), and 'The Reinhardt Conjecture as an Optimal Control Problem', authored by Thomas Hales ([Hal17]). Additionally, the code for the project is not yet fully completed (as of 7/9/2017), in that I am still attempting to reduce computational complexity. To this end, there may be trivial changes to functions (for the purposes of debugging and optimization), but the core functionality will remain intact.

1 Introduction

As noted above, this project is associated with the Reinhardt Conjecture, which hypothesizes that the shape of the centrally symmetric body in the plane that has the smallest optimal lattice-packing is the smoothed octagon. For our purposes, the <u>cost</u> of a shape will be the density of its optimal lattice-packing, up to a factor of $\sqrt{12}$ (the area of the hexagon which circumscribes the unit circle).
There is an infinite family of functions $\mathcal{F}$, each of which traverse
$U = \{(u_0, u_1, u_2) \in \mathbb{R}^3 | \sum_{i=0}^{2} u_i = 1, u_i \geq 0\}$ for some time t_f (t_f can vary for each

function $f \in \mathcal{F}$). Then for an arbitrary $f \in \mathcal{F}$, $f : [0, t_f] \to U$, where $f(t) = (u_0(t), u_1(t), u_2(t))$. We can then view cost as a map from $\mathcal{F}$ to $[0, \sqrt{12}]$, where the output is the proportion of space tiled by the optimal lattice packing of the shape corresponding to the function f (again, up to a factor of $\sqrt{12}$). The correspondence is detailed in Section 1.1.

The circle is given by the function $c(t) = (u_0(t), u_1(t), u_2(t)) \equiv (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, and $cost(c) = \pi$. Furthermore, we know that the circle corresponds to a saddle point in this space. Then there must exist functions in $\mathcal{F}$ whose cost is strictly less than π, and it is believed that finding such a cost (using bang-bang control) would correspond to a local, and potentially global, minimum. The purpose of the project is to utilize certain pieces of the math introduced in [Hal17] and find a hexagonally symmetric shape whose cost is strictly less than that of the circle. It would be desirable that the cost be a local minimum.

1.1 Control Simplex and Differential Equations

We begin by observing the 2-simplex, $U = \{(u_0, u_1, u_2) \in \mathbb{R}^3 | \sum_{i=0}^{2} u_i = 1, u_i \geq 0\}$. This is the set of controls for the problem, which correspond to the differential equations

$$x' = f_1(x, y; u) := \frac{y(b+2ax-cx^2+cy^2)}{b+2ax-cx^2-cy^2} \qquad (1)$$

$$y' = f_2(x, y; u) := \frac{2(a-cx)y^2}{b+2ax-cx^2-cy^2} \qquad (2)$$

by the map $Z_0 : U \to \mathfrak{sl}_2(\mathbb{R})$ such that $Z_0(u) = \begin{pmatrix} a & b \\ c & -a \end{pmatrix}$ (as defined in [Hal17]). We have that

$$a = \frac{u_1 - u_2}{\sqrt{3}}$$

$$b = \frac{u_0 - 2u_1 - 2u_2}{3}$$

$$c = u_0$$

Due to the bang-bang nature of the solution, we know that the solution of controls will consist exclusively of some combination of $u \in \{e_1, e_2, e_3\} \subset U$, with $e_1 = (1, 0, 0)$, $e_2 = (0, 1, 0)$, and $e_3 = (0, 0, 1)$. From here, we can obtain a simplified form of the differential equations, and specifically, for e_2 and e_3, we obtain

$$x' = y$$

$$y' = \frac{y^2}{m+x}$$

with $m = \pm\frac{1}{\sqrt{3}}$ (negative case for e_2 and positive case for e_3). As we are contained within the boundary of the star region $\mathfrak{h}^*$ given by $x = -\frac{1}{\sqrt{3}}$, $x = \frac{1}{\sqrt{3}}$, $x^2 + y^2 = \frac{1}{3}$, we can infer that $m + x \neq 0$. Solving the pair of differential equations yields

$$x(t) = -m + c_0 e^{c_1 t}$$

$$y(t) = c_0 c_1 e^{c_1 t}$$

where $(c_0, c_1) = (x_0 + m, \frac{y_0}{x_0+m})$, and (x_0, y_0) are the initial conditions $(x(0), y(0))$, which will thus define an initial complex value $z_0 = x_0 + iy_0$, with $i = \sqrt{-1}$. Note that with the given parametrization of x and y, we can express y as a function of x, with $y = c_1(x + m)$, illustrating that as the parameter t progresses from 0 to t_f (for a fixed control at e_2 or e_3), the path being traced in $\mathfrak{h}^*$ is a line with slope c_1 ($c_1 < 0$ for e_2 and > 0 for e_3).

Additionally, the e_1 control also yields an explicit solution, although the formulas for the solutions of $x(t)$ and $y(t)$ are significantly more complex than the previous two. For now, it is sufficient to note that the trajectories for e_1 control are circular arcs (with counter-clockwise orientation) that are symmetric about the imaginary axis, and each circular arc is a piece of a circle that passes through the points $(\pm\frac{1}{\sqrt{3}}, 0)$. Therefore, each point in $\mathfrak{h}^*$ has a unique circular arc that passes through it. Specifically, the centers of these circles are given by $(h, k) = (0, \frac{x^2+y^2-\frac{1}{3}}{2y})$, where $(x, y) \in \mathfrak{h}^*$ is a point on the circle, and the radius is $\sqrt{\frac{1}{3} + k^2}$. However, by taking advantage of the hyperbolic symmetry of $\mathfrak{h}^*$ (more on this in Section 3.1), we can eliminate the need for an explicit parametrization of these arcs.

To summarize, we have a control simplex which corresponds to differential equations whose solutions define trajectories in the complex plane. As the correct control $f(t)$ will necessarily be bang-bang, we need only concern ourselves with trajectories that consist of the previously defined lines and circular arcs. Then the correct control will satisfy the conditions for an admissible trajectory (as defined in [Hal17]), which has an associated hexagonally symmetric disk. This disk is given by acting a matrix $g \in SL_2(\mathbb{R})$ on $e_j^* = (\cos(\frac{2\pi j}{6}), \sin(\frac{2\pi j}{6})), j = 0, 1, ..., 5$ by matrix multiplication on each of the six vectors. The paths traced out here in $\mathbb{R}^2$ thus define the boundary of the hexagonally symmetric disk. The cost of this disk will necessarily be less than π (assuming it is a solution to this project). Lastly, note that the explicit formula for the cost can be given by

$$\int_0^{t_f} \frac{x^2 + y^2 + 1}{y} dt$$

, as stated by (18) in [Hal17].

In order to obtain g, we use the information from the path in $\mathfrak{h}^*$ traced out by the e_3 trajectory. Even if we are not on an e_3 trajectory, we can use the hyperbolic symmetry to obtain an approriately corresponding e_3. We use this information to obtain $g(s)$ (which is not the same as the matrix g, but rather is $\gamma_0(z,t)$ when s is replaced with $e^{c_1 t}$) as calculated in the code by 'gs', and compute the appropriate $\gamma_i = R^i \gamma_0 R^{-i}$ functions needed to evaluate g, where $R \in SO_2(\mathbb{R})$ is $\begin{pmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}$, the rotation matrix by $\frac{\pi}{3}$. The takeaway message here is that the solutions to the above differential equations (including the initial conditions), along with the set of controls do indeed provide the needed information to obtain g, and thus, the hexagonally symmetric disk $D(g)$.

2 Control Trajectory

In order to find a cost less than that of the circle, we want to deform the circle in some hexagonally symmetric manner, and obtain the new cost. First, we observe that the solutions to (1) and (2) for the circle are
$x \equiv 0$, $y \equiv 1$
giving us the constant solution of $z(t) \equiv i$, where $z(t) = x(t) + iy(t)$ takes on values in $\mathfrak{h}^*$. Therefore, in order to deform the circle, we should choose some function $f \in \mathcal{F}$ such that the corresponding solutions to (1) and (2) yield a trajectory in $\mathfrak{h}^*$ that both starts and ends at $z = i$. See Figure 1 as an example. Here, we start at $z_0 = 0 + 1i$ (as we must), and we travel along an e_3 trajectory for some time t_1, which is our first switching time, where we switch control to e_1 for some time t_2. Then at time $T_2 = t_1 + t_2$ we have our second switching time, where we switch to control e_2 for time t_3, etc. Note that we end at $0 + 1i$ again, as we must, in order to deform the circle in some hexagonally symmetric manner. It is with these notions in mind (starting and ending at i, using only e_1, e_2, and e_3 trajectories) that we start the search for the function that will yield a cost below π.

2.1 Simple Control Trajectory

Since we have now seen that functions generate these trajectories in $\mathfrak{h}^*$, we can refer to a trajectory without explicitly referencing the function that generates it. However, we still implicitly understand there is a unique function in $\mathcal{F}$ that generates the trajectory. Therefore, we can reference a particular trajectory and the corresponding function interchangeably. We will primarily reference the trajectory, rather than the function.

It should be noted that while we can compute the cost for any control trajectory we would like (provided it corresponds to a closed disk), the easiest to work with computationally are elementary trajectories. First, we will focus on a particular family of trajectories, called simple control trajectories. This type is of the form as illustrated in Figure 2, in which we start with an e_3 trajectory for some time t_1, followed by an e_1 trajectory, in which the x-coordinate at $x(t_1)$ is reflected across the imaginary axis, finally followed by an e_2 trajectory which travels back to $z_0 = i$. We should note that everything about a simple control trajectory is determined by choosing the $x_1 = x(t_1)$ position. We can use the equation of the line to obtain y_1, and then symmetrically reflect across the imaginary axis to obtain $(x_2, y_2) = (-x_1, y_1)$. We can also obtain the times t_1, t_2,and t_3 by using the parametrization of the x (or y) coordinate. Again, we take advantage of the hyperbolic symmetry to compute t_2, rather than solving for t_2 given the e_1 parametrization. See the code for details.

There is an associated function 'error' in the code which is used to determine if the hexagonally symmetric figure constructed from simple control trajectory functions is closed (it can alternatively be six disjoint pieces; connectedness is not guaranteed for an arbitrary control trajectory). Details regarding the error function will be given in Section 3.3, but for now, it should be mentioned that the error function for simple control trajectories does not evaluate to 0 except for the circle function $c(t)$ (a special case of the simple control trajectory), which leads us to believe that there do not exist any (nontrivial) simple control trajectories whose error is 0. Thus there do not appear to be any hexagonally symmetric, closed disks generated by simple control trajectories, meaning that there are also no costs to compute either. The solution does not appear to exist as a simple control trajectory, and while this has certainly not been proven, computational experiments seem to yield this conclusion. Thus the next stage of the project required us to create different (but still relatively elementary) control trajectories.

2.2 Two and Three Parameter Control Trajectories

As noted above, the simple control trajectories were completely determined by the single parameter x_1. These are not the only types of control trajectories that are determined by one parameter, but we chose to create trajectories which depended on multiple parameters. An insight during the project led us to believe that a single parameter may not be 'enough', in the sense that more degrees of freedom gives us more control over the trajectory. We could instead search through a two or three dimensional space of parameters to yield the desired result, rather than the one-dimensional space offered to us through the simple control trajectories. Thus the two and three parameter trajectories were created. Note that again these are not unique types of trajectories for two or three parameters, but rather were chosen in such a way so as to minimize algorithmic efforts to determine the entire trajectory. See Figures 3 and 4 for illustrations, along with the code for details. In particular, the three-parameter trajectory is considered to be especially desirable, not only because it utilizes more parameters, but because it takes full advantage of the hyperbolic symmetry at play, leading our intuition to believe that a solution exists within these parameters for this type of control trajectory, due to the seemingly maximal amount of influence we have on the outcome. This will be elaborated upon in Section 3.1.

3 Underlying Mathematics

Here, we will go into more mathematical depth regarding certain functions in the code. In particular, we will discuss the math related to the three-parameter control trajectory (in the code, this is the condition of len(args) == 3), the 'bar' function, and the 'error' function.

3.1 Hyperbolic Symmetry

As has been referenced throughout the summary thus far, there is some hyperbolic symmetry at play which we are taking advantage of to greatly simplify various computations. To be more explicit, we are using the group action of the linear fractional transformation on the upper half-plane (in particular, $\mathfrak{h}^*$). We have $\varphi\colon SL_2(\mathbb{R}) \times \mathfrak{h} \longrightarrow \mathfrak{h}$ such that $\varphi(m,z) = m.z$ where . denotes the action, and $\mathfrak{h} = \{x+iy \in \mathbb{C} | y \geq 0\}$. If $m = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(\mathbb{R})$, then $m.z = \frac{az+b}{cz+d}$. In particular, we use $m = R$, with R defined as above. Here we have that $R.\frac{1}{\sqrt{3}} = -\frac{1}{\sqrt{3}}$, $R.-\frac{1}{\sqrt{3}} = \infty$, and $R.\infty = \frac{1}{\sqrt{3}}$ (we have appended ∞ to $\mathfrak{h}$ under this transformation, treating the line $y = \infty$ as a single point).

Now, let us observe that the boundary of $\mathfrak{h}^*$ is a hyperbolic triangle, with vertices $(\pm\frac{1}{\sqrt{3}}, 0)$ and ∞. Then the significance of R in this context is that it acts on the triangle by rotation, rotating the triangle with 'clockwise' orientation. It is then clear to see that R has order 3 on the vertices of $\mathfrak{h}^*$. Yet it is not quite as obvious that

R actually has order 3 on the entirety of $\mathfrak{h}$, and in-particular on $\mathfrak{h}^*$, given that R is a rotation matrix of $\frac{\pi}{3}$, one sixth of a circle. However, observe that $R^3 = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$, and $R^3.z = \frac{-z+0}{0-1} = z$, illustrating that R^3 is in the kernel of the action. In other words, R has order 3 when acting on any (fixed) $z \in \mathfrak{h}$.

Given the order 3 symmetry, we can (after tedious amounts of computation) observe that under the linear fractional transformation by R, e_3 trajectories map to e_2 trajectories, which map to e_1 trajectories, which then map back to the original e_3 trajectories. The significance this bears for us then, is that we can transform simple control trajectories into the separate components that compose our three-parameter control trajectories. Specifically, we can transform a simple control trajectory into a 'right' trajectory by acting R on it, and similarly for a 'left' trajectory by acting $R^{-1} = R^2$ on it. See Figure 5 for visual intuition regarding the three-parameter control trajectory. It is due to the complete incorporation of all three symmetries of $\mathfrak{h}^*$ that one might believe a solution may be found with three parameters.

3.2 Continuity and Differentiability of g

3.2.1 continuity

The matrix $g \in SL_2(\mathbb{R})$, as determined by a given control trajectory, can be easily constructed to be continuous at switching times of the control (g must be continuous if it can construct a continuous disk). Note that it is necessary for an admissible trajectory to correspond to g such that $g(0) = I_{2\times 2}$, the identity matrix in $SL_2(\mathbb{R})$. Given that g is constructed from multiplication of various γ_i functions, one need only shift the time parameter by the current switching time to obtain continuity, as $\gamma_m(z, t = 0) = I_{2\times 2}$, for any $m \in \mathbb{Z}$ and any initial condition $z \in \mathbb{C}$.

In general, if $t \in [T_i, T_{i+1}] \subset [0, t_f]$, then
$g(t) = \gamma_{k_0}(\bar{z_0}, t_1)\gamma_{k_1}(\bar{z_1}, t_2)\gamma_{k_2}(\bar{z_2}, t_3)\cdots\gamma_{k_{i-1}}(\bar{z_{i-1}}, t_i)\gamma_{k_i}(\bar{z_i}, t - T_i)$, where $T_{j+1} = T_j + t_{j+1}$ for $j = 0, 1, 2, \ldots i$ (i.e. $T_j = t_1 + t_2 + \cdots + t_j$, as $t_0 = 0$), and $k_j \in \mathbb{Z}$. It is not necessarily the case that $\bar{z_j} = z_j$; this notation will be explained when discussing differentiability. Note that as R has order 3, we can compute k_j modulo 3. Additionally, note that the variable t only appears in the $i + 1^{st}$ (the last) gamma function; the previous i gamma functions are constants. It is the combination of the inclusion of these constants, along with the shift by T_i in the final gamma function that guarantees continuity of g.

3.2.2 differentiability

As given in [Hal17], $g' = gX$, where $'$ indicates the derivative,
and $X_z = \begin{pmatrix} \frac{x}{y} & -\frac{x^2+y^2}{y} \\ \frac{1}{y} & -\frac{x}{y} \end{pmatrix} = \hat{z}J\hat{z}^{-1}$, with $J = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ and $\hat{z} = \begin{pmatrix} y & x \\ 0 & 1 \end{pmatrix}$ where $z = x + iy$. (Additionally, it is helpful to view $\hat{}$ as an injective function mapping z to $\hat{z}$.) So the issue is not with differentiability itself, but rather with the continuity of g', and in particular, with continuity at the switching times. In general, for $t \in [T_i, T_{i+1}]$ such that g is defined as before, then
$g'(t) = \gamma_{k_0}(\bar{z_0}, t_1)\gamma_{k_1}(\bar{z_1}, t_2)\cdots\gamma_{k_{i-1}}(\bar{z_{i-1}}, t_i)R^{k_i}\gamma_0(\bar{z_i}, t - T_i)X_{e_3(\bar{z_i}, t-T_i)}R^{-k_i}$,
where $e_3(z,t) = -\frac{1}{\sqrt{3}} + c_0e^{c_1t} + i(c_0c_1e^{c_1t})$ is the complex number obtained when traveling on an e_3 trajectory for time t with initial condition $z = x_0 + y_0 i$. c_0 and c_1 are then determined as given in Section 1.1 above. Then continuity (of g') is clear, with the exception of switching times. Thus we would like to show continuity of $g'(T_i)$. Before doing this, however, it is best to present a useful lemma.

Lemma 1. *For any z, $\bar{z} \in \mathfrak{h}^*$, if $\hat{z}J\hat{z}^{-1} = R^i\hat{\bar{z}}J\hat{\bar{z}}^{-1}R^{-i}$, then $\bar{z} = R^{-i}.z$, where J, R, $\mathfrak{h}^*$, and $\hat{}$ are as previously defined, '.' denotes the linear fractional transformation, and $i \in \mathbb{Z}$.*

Proof. First, note that $C_{SL_2(\mathbb{R})}(J) = SO_2(\mathbb{R})$. Then if $g_1Jg_1^{-1} = g_2Jg_2^{-1}$ for any g_1, $g_2 \in SL_2(\mathbb{R})$, we have $g_1SO_2(\mathbb{R}) = g_2SO_2(\mathbb{R})$ $(*)$.
This is true since $g_1Jg_1^{-1} = g_2Jg_2^{-1} \Leftrightarrow g_1^{-1}g_2Jg_2^{-1}g_1 = J \Leftrightarrow g_1^{-1}g_2 \in C_{SL_2(\mathbb{R})}(J) = SO_2(\mathbb{R}) \Leftrightarrow g_2 \in g_1SO_2(\mathbb{R}) \Leftrightarrow g_2SO_2(\mathbb{R}) = g_1SO_2(\mathbb{R})$.
Second, we have that $R^i\hat{\bar{z}}J\hat{\bar{z}}^{-1}R^{-i} = \widehat{R^i.\bar{z}}J\widehat{R^i.\bar{z}}^{-1}$, thus $\hat{z}SO_2(\mathbb{R}) = \widehat{R^i.\bar{z}}SO_2(\mathbb{R})$ by fact $(*)$ above. Therefore, $\hat{z} \in \widehat{R^i.\bar{z}}SO_2(\mathbb{R})$, so $\hat{z} = \widehat{R^i.\bar{z}} * r$ for some $r \in SO_2(\mathbb{R})$. However, note that $\hat{z} = \hat{z} * I_{2\times 2}$, with $I_{2\times 2} \in SO_2(\mathbb{R})$. By the Iwasawa decomposition, $\hat{z} * I_{2\times 2} = \widehat{R^i.\bar{z}} * r \Rightarrow \hat{z} = \widehat{R^i.\bar{z}}$ (and $I_{2\times 2} = r$). Since the $\hat{}$ operation is one-to-one, we have that $z = R^i.\bar{z}$. The result follows. ∎

In order to obtain continuity at $t = T_i$, let us determine g' first when viewing T_i as belonging to $[T_{i-1}, T_i]$, and second when viewing it as belonging to $[T_i, T_{i+1}]$. In the first case,
$g'(T_i) = \gamma_{k_0}(\bar{z_0}, t_1)\gamma_{k_1}(\bar{z_1}, t_2)\cdots\gamma_{k_{i-2}}(\bar{z_{i-2}}, t_{i-1})R^{k_{i-1}}\gamma_0(\bar{z_{i-1}}, t_i)X_{e_3(\bar{z_{i-1}}, t_i)}R^{-k_{i-1}}$. In the second case,
$g'(T_i) = \gamma_{k_0}(\bar{z_0}, t_1)\gamma_{k_1}(\bar{z_1}, t_2)\cdots\gamma_{k_{i-1}}(\bar{z_{i-1}}, t_i)R^{k_i}\gamma_0(\bar{z_i}, T_i - T_i)X_{e_3(\bar{z_i}, T_i-T_i)}R^{-k_i}$. These two values must be equal to each other to achieve continuity, so by equating case 1 with case 2, performing algebraic simplifications, and setting $\tilde{z} = e_3(\bar{z_{i-1}}, t_i)$ we

obtain that $X_{\tilde{z}} = R^{k_i - k_{i-1}} X_{\bar{z_i}} R^{k_{i-1} - k_i}$. Expanding X as previously given, we obtain $\hat{\tilde{z}} J \hat{\tilde{z}}^{-1} = R^{k_i - k_{i-1}} \hat{\bar{z_i}} J \hat{\bar{z_i}}^{-1} R^{-(k_i - k_{i-1})}$. By the lemma, we then have that $\bar{z_i} = R^{k_{i-1} - k_i}.\tilde{z} = R^{k_{i-1} - k_i}.e_3(\bar{z_{i-1}}, t_i)$. This then illustrates the need and construction of the 'bar' function of the code. Additionally, given the context of our situation where $z_0 = i$, it is true that $R^j.z_0 = z_0$ for any j. Then given any particular control trajectory, we have shown that 'g' (as defined in the code) is continuously differentiable.

3.3 Closure

Now that we can confirm g is continuously differentiable, we need only to show that it generates a closed shape. Continuity of g' is necessary, but not sufficient for the hexagonally symmetric shape to be closed. This fact then motivates the 'error' function in the code. The 'error' simply returns $(a-d)^2 + (b+c)^2$, where $g = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, which will evaluate to 0 iff the shape is closed.

The idea is that for a given g, each hexagonally symmetric piece of the generated shape must deviate from the corresponding hexagonally symmetric piece of the unit circle, and then return back to it at time t_f. This deviation is precisely the deformation of the circle we want.

For the unit circle, with $z(t) \equiv i$, we have $g(t) = \begin{pmatrix} cos(t) & -sin(t) \\ sin(t) & cos(t) \end{pmatrix}$. In this case, g is a homomorphism from $\mathbb{R}$ to $SO_2(\mathbb{R})$ such that $g(s)g(t) = g(s+t)$, for $s, t \in \mathbb{R}$ (mod 2π, if you'd like). In particular, as $R = g(\frac{\pi}{3})$, then $R^j = g(j\frac{\pi}{3})$. Furthermore, $e_j^* = R^j e_0^*$. Note that we can no longer take j mod 3, as the action is now multiplication rather than the Mobius transformation. Here, R has order 6. Then the σ_j curve will be given by $\sigma_j(t) = g(t)e_j^* = g(t + j\frac{\pi}{3})e_0^* = \begin{pmatrix} cos(t + j\frac{\pi}{3}) & -sin(t + j\frac{\pi}{3}) \\ sin(t + j\frac{\pi}{3}) & cos(t + j\frac{\pi}{3}) \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} cos(t + j\frac{\pi}{3}) \\ sin(t + j\frac{\pi}{3}) \end{pmatrix}$. Thus we can now parametrize the circle in terms of σ_j, using all j.

Now for closure of a particular shape, we must have that $\sigma_j(t_f) = \begin{pmatrix} cos(t' + j\frac{\pi}{3}) \\ sin(t' + j\frac{\pi}{3}) \end{pmatrix}$ for some particular time $t'(**)$. (Note t' is not a derivative here.) See Figure 6 for reference.

Let g_{shape} refer to the function g generated by some (not necessarily closed) shape, and let g_{circle} be analogously defined for the unit circle. Then condition $(**)$ holds iff $(\forall j)\ g_{shape}(t_f)e_j^* = g_{circle}(t')e_j^*$ (due to the hexagonal symmetry). Given that (a subset of) $\{e_j^*\}$ forms a basis, this is true iff $g_{shape}(t_f) = g_{circle}(t') \Leftrightarrow g_{shape}(t_f) \in SO_2(\mathbb{R}) \Leftrightarrow a = d$ and $b = -c \Leftrightarrow (a-d)^2 + (b+c)^2 = 0$. This then justifies the 'error' function. Thus the search for the hexagonally symmetric deformation of the circle whose cost is less than π can be reduced to the search for the error function evaluating to 0 (or within a machine epsilon of 0, due to rounding). The error function accepts a 1, 2, or 3-tuple, generating that tuple's control trajectory and g, so the true solution shall be the shape generated by the corresponding σ_j functions.

4 Concluding Remarks

As you may have already concluded due to the lack of the presentation of a solution, I have not yet (as of 7/9/17) found a solution to the problem. Furthermore, it is not clear to me whether a solution via these methods is guaranteed to exist. We know, given that the circle corresponds to a saddle point, that there is a solution, yet I am not sure if we can conclude that the solution will actually be bang-bang. Of course it has been proven that various solutions will be (such as the smoothed octagon), but I cannot be certain that the prescribed method can account for this. For instance, what if there are not finitely many switching times for any of the circle deformations? What's more, we are only accounting for a very small subset of all possible bang-bang controls, decreasing the likelihood of finding a solution. However, these are worst-case scenario speculations, and given the explanation provided in 3.1, I am reasonably hopeful that a solution can be found with the three parameter control trajectory. It truly does capture all the symmetry of the problem, and gives us maximal control over the outcome. Additionally, there are plenty of solutions to the problem, so while we may be searching for a needle in the haystack, the good news is that there are lots of needles.

Potential future use of the code could be to iterate through many different control parameters to attempt to find a solution, or to create entirely new control trajectories to utilize in the search. Personally, I plan to perform an iterated search through 3-tuples, and make a second attempt at using a built-in minimization function. I don't currently intend to create more control trajectories, seeing as one could continue perpetually creating new ones to no avail, but if readers have their own intuition towards what may work, they are free to download and use the code however they wish.

4.1 Personal Comments

Lastly, I would like to make some personal remarks regarding the project. For me, this was a new experience in many regards. I have never before made legitimate attempts at reading mathematics at the level of that which is published in peer-reviewed journals. I have never before undertaken a programming project of such scope, requiring large amounts of time and sufficient mathematical background. On that note, I have never before incorporated higher-level mathematics into my programs at all. All of these things are now true due to this project.

In addition to these new and enjoyable experiences, there have also been some new experiences that have proven quite frustrating. The main negative recurring theme throughout the project was issues with Sage functionality, in-particular with respect to coercion. There were (and continue to be) multiple occasions for which a function would be written, but the underlying Sage operations would misinterpret the inputs, and be unable to perform the desired actions, despite the fact that the code was seemingly logically sound. In general, one must be very careful to ensure that various data types play well together. In my case, I most often had coercion issues between type real field (with the standard 53 points of precision) and type Expression. Often, unbeknownst to me, built-in Sage code would only accept one or the other, leaving me pondering these issues when confronted with error messages, as it is visually difficult to identify the difference. Very slowly -and with a lot of help- I have managed to understand these issues to a greater degree. However, as it stands, similar coercion issues are still what are preventing me from finding a local minimum to the 'error' function, and quickly determining a potential solution.

All the same, though these issues have caused significant time inefficiencies and frustration, they have still presented a learning opportunity, through the need for meticulously tracking what is happening to the software's code. I have also come to appreciate that although the ideal for software is that it runs smoothly enough to be treated as a black box, it is not always possible. This has illustrated to me the need to be capable of understanding what is going on under the hood. Additionally, these experiences have given me new academic directions I would like to pursue to some reasonable extent; primarily category theory and topology. In this regard, I would like to thank Jacob Gross for his willingness to discuss his own work, and spend time illuminating concepts related to his own research on the Reinhardt Conjecture. Through these discussions, he has helped provide me with said academic directions. Furthermore, due to his level of mathematical knowledge and enthusiasm to provide explanations, he has helped me to contextualize broad branches of mathematics in a way I would currently be unable to reciprocate.

Finally, I would like to thank Dr. Thomas Hales for his continued support throughout this project. He has spent many hours guiding me through the process, and has allowed me to overcome numerous roadblocks that I certainly would not have navigated alone. His insight and guidance were critical to the progress that has been made, and it would be difficult to overstate his contribution. I am extremely grateful for his instruction.

5 Glossary of Standard Notation

1. $\mathbb{R}$: the field of real numbers
2. $\mathbb{C}$: the field of complex numbers
3. $\mathfrak{h}$: $\{x+iy \in \mathbb{C} | y > 0\}$, the upper half-plane of complex numbers
4. $\mathfrak{h}^*$: $\{x+iy \in \mathfrak{h} | -\frac{1}{\sqrt{3}} < x < \frac{1}{\sqrt{3}}, x^2+y^2 > \frac{1}{3}\}$, sometimes referred to as the star region
5. $SL_2(\mathbb{R})$: the special linear group of 2×2 matrices with entries in $\mathbb{R}$
6. $\mathfrak{sl}_2(\mathbb{R})$: the Lie algebra of $SL_2(\mathbb{R})$, whose elements have trace 0
7. $SO_2(\mathbb{R})$: the special orthogonal group of 2×2 matrices with entries in $\mathbb{R}$
8. $I_{2\times 2}$: the 2×2 identity matrix $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$