2.1

Non-Euclidean Geometry

Question 1

You have,

$$|z_1|^2 - \bar{a}z_1 - a\bar{z_1} + |a|^2 = |a|^2 + 1$$
$$|z_2|^2 - \bar{a}z_2 - a\bar{z}_2 + |a|^2 = |a|^2 + 1$$

So solving for a you get,

$$a = \frac{z_2(1 - |z_1|^2) - z_1(1 - |z_2|^2)}{\bar{z}_2 z_1 - z_2 \bar{z}_1}$$

The program which given $z_1, z_2, z_3 \in \mathbb{C} \cup \{\infty\}$, draws and fills the spherical triangle between them is given on page 9 and 10. The following diagrams show the output in a few cases. Note that when one of the vertices is ∞, the spherical triangle is an unbounded region as shown.

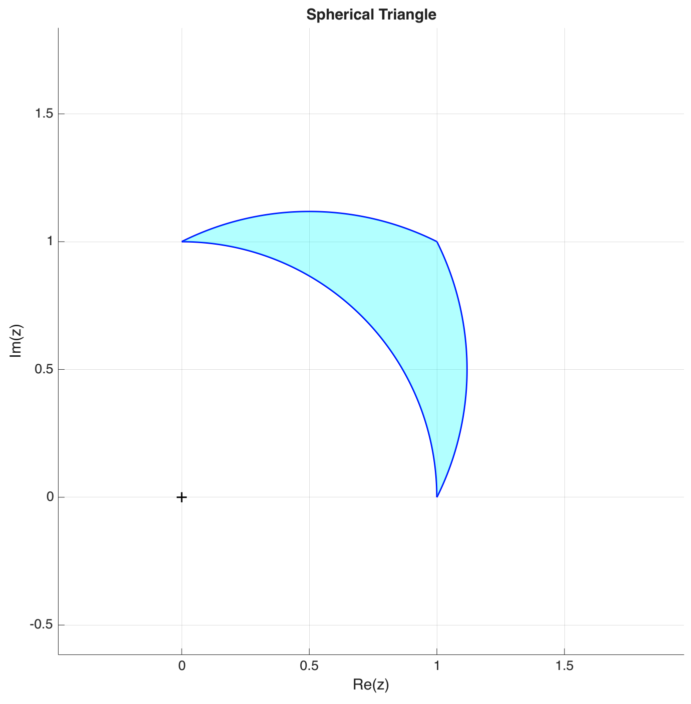

Figure 1: Spherical triangle with vertices at $1, i, 1+i$

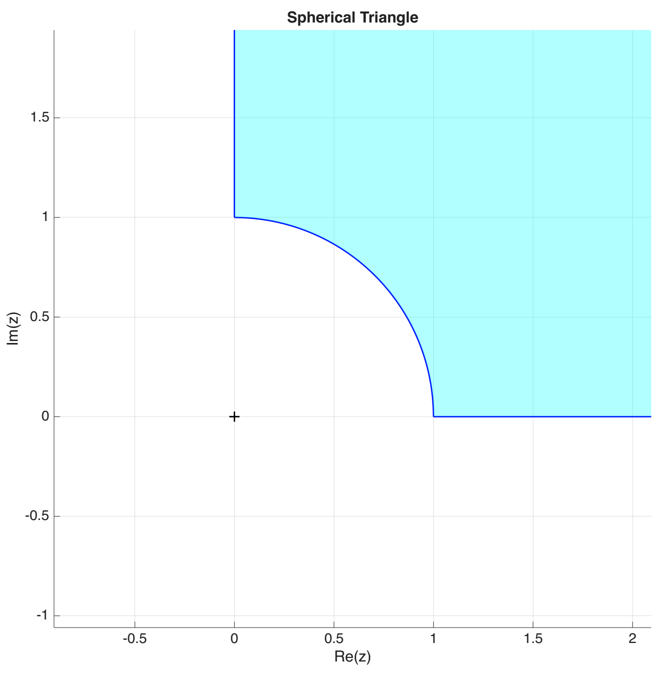

Figure 2: Spherical triangle with vertices $1, i, \infty$

Note that the intersection points of a spherical line and the unit circle are invariant under stereographic projection. On the unit sphere, these are the intersection of a great circle and the equator, i.e. a pair of antipodal points on the unit circle.

Question 2

The program to draw and fill the hyperbolic triangle between 3 points is given on page 11. Below are a few cases of the output (the boundary circle is also shown).

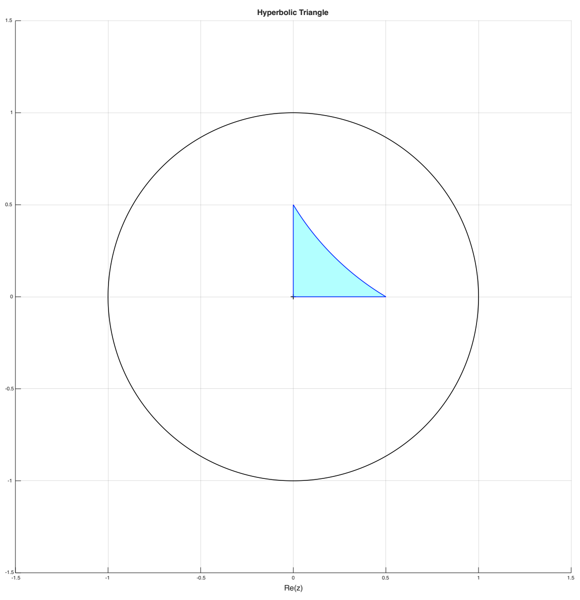

Figure 3: Hyperbolic triangle with vertices $0, 1/2, i/2$

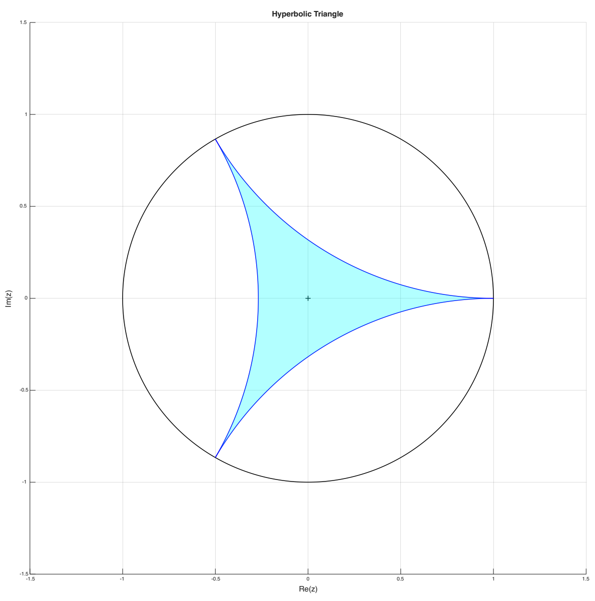

Figure 4: Hyperbolic triangle with ideal vertices as the 3rd roots of unity

Given a hyperbolic line ℓ and a point $z_0 \in \ell$ by using a Möb(D) transformation (specifically $z \mapsto \frac{z-z_0}{1-\bar{z}_0 z}$) we map ℓ to a diameter which clearly intersects the unit circle orthogonally. Since Möb(D) preserves the unit circle and Möbius transformations are conformal, ℓ must also intersect the unit circle orthogonally. We can also see this in Figure 4.

Question 3

Given a spherical triangle with internal angles $\alpha_1, ..., \alpha_n$, we can subdivide our spherical n-gon into n spherical triangles, by picking a point in the interior and drawing n spherical lines between this point and the vertices. Noting that the area of a spherical triangle with internal angles α, β, γ is $\alpha + \beta + \gamma - \pi$, we obtain,

$$\text{Area of spherical } n\text{-gon} = \sum_{i=1}^{n} \alpha_i + 2\pi - n\pi = \sum_{i=1}^{n} \alpha_i - (n-2)\pi$$

Therefore, the area of a spherical n-gon with all internal angles $2\pi/3$ is $\frac{\pi}{3}(6-n)$, which must be strictly positive, so $n < 6$. So $n = 3, 4, 5$, and now we show that in all of these cases such an r does exist.

Let the spherical n-gon with vertices at the roots of $z^n = r^n$ be P_r. We see that the internal angles of P_r are all equal so it suffices to find an r such that P_r has the required area. Note that for $0 \leq r \leq 1$, Area(P_r) is increasing and continuous with Area$(P_0) = 0$ and Area$(P_1) = 2\pi$.

Since $\frac{\pi}{3}(6-n) \in (0, 2\pi)$, by IVT we conclude that such an r exists in $(0, 1)$.

For a hyperbolic polygon with internal angles $\alpha_1, ..., \alpha_n$, as in the spherical case, we subdivide into hyperbolic triangles and use the fact that the area of a hyperbolic triangle with internal angles α, β, γ is $\pi - (\alpha + \beta + \gamma)$ to get,

$$\text{Area of hyperbolic } n\text{-gon} = (n-2)\pi - \sum_{i=1}^{n} \alpha_i$$

So, the area of a hyperbolic n-gon with all internal angles $2\pi/3$ is $\frac{\pi}{3}(n-6)$ and so we see that $n \geq 7$.

As before, let the hyperbolic n-gon ($n \geq 7$) with vertices at the roots of $z^n = r^n$ be Q_r. The ideal n-gon Q_1 has all internal angles 0, so has area $(n-2)\pi$. Since $\frac{\pi}{3}(n-6) \in (0, (n-2)\pi)$, and Area$(Q_r)$ is monotonically increasing and continuous, we conclude that the required r exists.

We can clearly subdivide P_r into n spherical triangles by taking the spherical lines between 0 and all the vertices. The resulting spherical triangles then have internal angles $2\pi/n, \pi/3, \pi, 3$, and are symmetric about the radial line which bisects the angle $2\pi/n$, so can be subdivided further into 2 spherical triangles with internal angles $\pi/n, \pi/3, \pi/2$ (since the bisecting line is orthogonal to the arc opposite the angle $2\pi/n$). This is our required subdivision.

We now compute the vertices of this subdivision using the spherical cosine rule, and plot the resulting subdivisions below for $n = 3, 4, 5$.

Consider the spherical triangle in the subdivision with two vertices at $0, r$ and such that the 3rd vertex has argument π/n, call this vertex ω. It suffices to find $|\omega|$ and r, since we know the vertices are at $re^{2ik\pi/n}$, $|\omega|e^{(2k-1)i\pi/n}$.

Noting that $\cos(2\tan^{-1}|z|) = \frac{1-|z|^2}{1+|z|^2}$, applying the spherical cosine formula to find r,

$$\left(\frac{1-r^2}{1+r^2}\right)\sin(\pi/n)\sin(\pi/3) = \cos(\pi/n)\cos(\pi/3)$$

$$\text{So} \quad r^2 = \frac{\sqrt{3}\tan(\pi/n) - 1}{\sqrt{3}\tan(\pi/n) + 1}$$

Likewise with $|\omega|$,

$$\left(\frac{1-|\omega|^2}{1+|\omega|^2}\right)\sin(\pi/n) = 1/2$$

$$\text{So} \quad |\omega|^2 = \frac{2\sin\pi/n - 1}{2\sin\pi/n + 1}$$

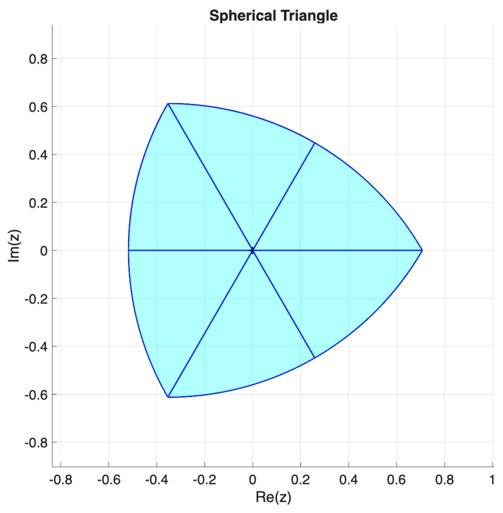

Figure 5: Subdivision for $n = 3$

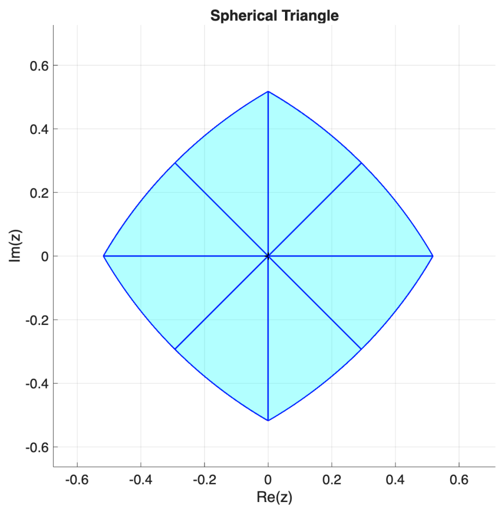

Figure 6: Subdivision for $n = 4$

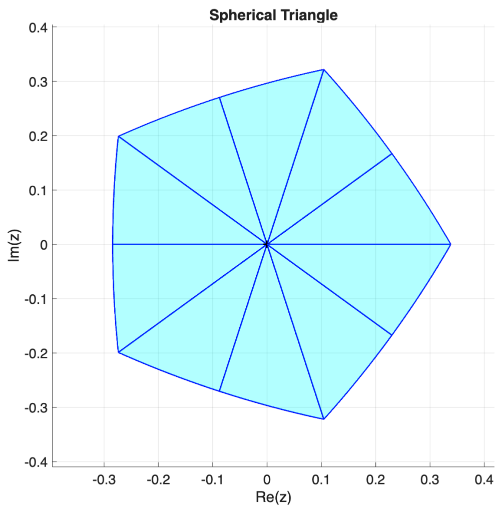

Figure 7: Subdivision for $n = 5$

For the hyperbolic case, the vertices of the subdivision have arguments $k\pi/n$ $(k = 0, 1, ..., 2n-1)$ as in the spherical case.

So considering the triangle with vertices at $0, r, \omega$ (as before), we compute $|\omega|$ and r using the hyperbolic cosine rule.

Noting that $\cosh(2\tanh^{-1}|z|) = \frac{1+|z|^2}{1-|z|^2}$ we obtain,

$$r^2 = \frac{1-\sqrt{3}\tan(\pi/n)}{1+\sqrt{3}\tan(\pi/n)}$$

Likewise,

$$|\omega|^2 = \frac{1-2\sin\pi/n}{1+2\sin\pi/n}$$

We plot the subdivisions for $n = 7, 8$ below.

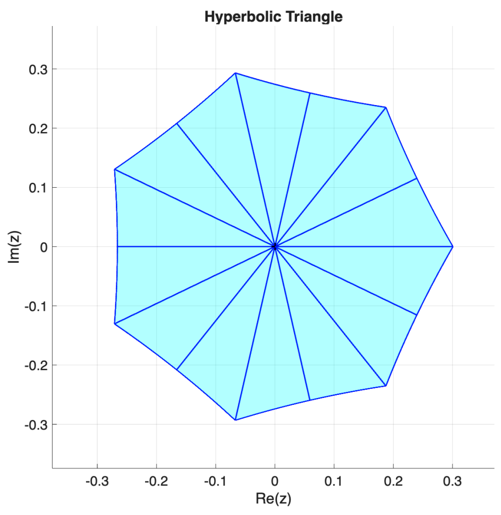

Figure 8: Subdivision for $n = 7$

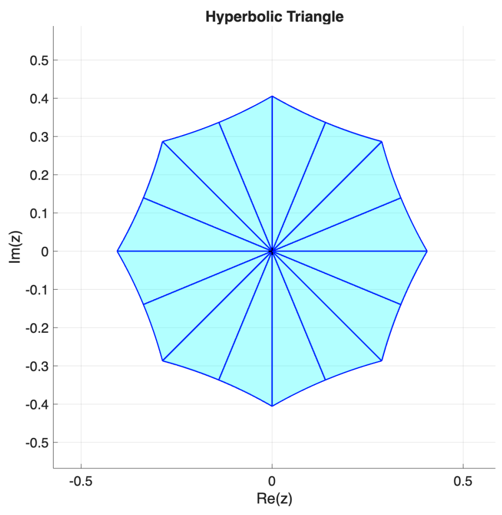

Figure 9: Subdivision for $n = 8$

Question 4

Firstly note that the composition of two reflections is a Möbius transformation. Since Möbius transformations are determined by the images of 3 points, it suffices to consider where the vertices of Δ_0 are mapped to by the S_i.

Fix Δ_0 to be the triangle with vertices at $0, r, \omega$. Then, we see that S_1 fixes ω, and maps $0, r$ on the spherical/hyperbolic line connecting them and ω, opposite ω whilst maintaining the spherical/hyperbolic distance to ω (as S_i are isometries). This corresponds to a (geodesic) rotation by angle π, and on the unit sphere S_1 is a rotation by π on the axis through $\pi_+^{-1}(\omega)$ (where π_+ is the standard stereographic projection map).

Likewise, S_2 fixes r while rotating Δ_0 an angle $2\pi/3$ clockwise about r. Therefore, this is a rotation by angle $2\pi/3$ clockwise about r and corresponds to a clockwise rotation by $2\pi/3$ on the axis through $\pi_+^{-1}(r)$ on the unit sphere (in the spherical case).

S_3 fixes the origin and rotates Δ_0 by angle $2\pi/p$ clockwise about the origin, so it is a clockwise rotation by angle $2\pi/p$ about the origin.

From this we can clearly see that S_1, S_2, S_3 have orders $2, 3$ and p respectively. Now note,

$$\sigma_1^2 = \pm I$$

$$\sigma_2^3 = \pm I$$

$$\sigma_3^p = \pm \begin{pmatrix} 1 & p \\ 0 & 1 \end{pmatrix} = \pm I$$

So we see that S_i has the same order as σ_i.

Question 5

The programs to compute admissible pairs (for $p = 3, 5, 7$) and plot the resulting triangles are listed on pages 13-19. The programs use the generators listed to compute elements of $\mathrm{PSL}(2, p)$ and then apply the corresponding transformations to Δ_0 to obtain admissible pairs.

We compute the order of $\mathrm{PSL}(2, p)$ (justifying the target count in the programs):

Note that the order of $\mathrm{GL}(2, p)$ is $(p^2 - 1)(p^2 - p)$ since given a matrix $A \in \mathrm{GL}(2, p)$, there are $p^2 - 1$ choices for the first column of A, and $p^2 - p$ choices for the second column of A (since it cannot be in the span of the first column). Now, noting that

$$\det : \mathrm{GL}(2, p) \to \mathbb{F}_p^\times$$

is surjective, you get $|\mathrm{SL}(2, p)| = \frac{(p^2-1)(p^2-p)}{p-1} = p^3 - p$

Now since $p \neq 2$, $|\mathrm{PSL}(2, p)| = (p^3 - p)/2$

For $p = 3$ we get the following plot:

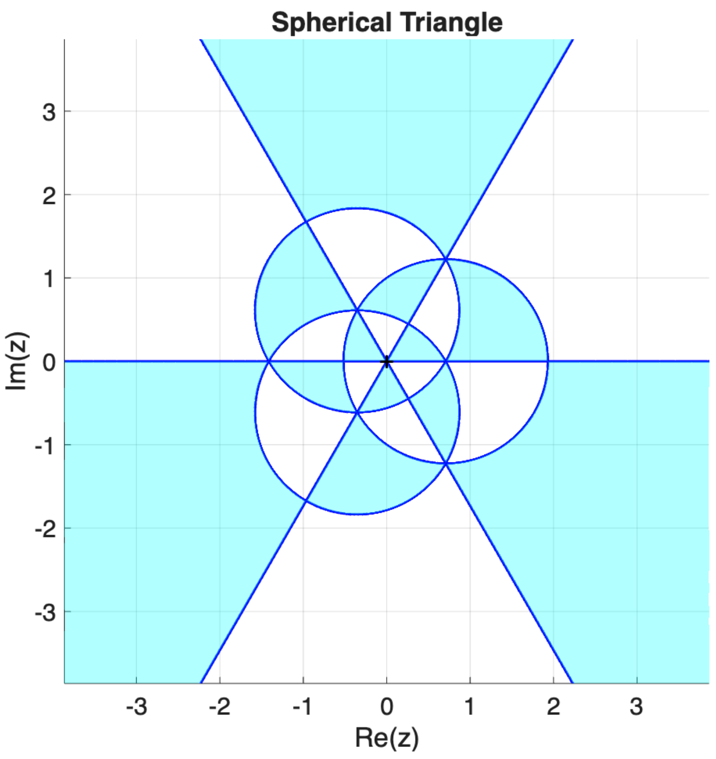

Figure 10: Admissible triangles for $p = 3$

We see there are 12 triangles and they never overlap. Since we have a tessellation of $\mathbb{C} \cup \{\infty\}$ and the S_i preserve orientation, we cannot obtain any new triangles by applying the S_i (since the isometries to obtain new triangles would have to be orientation reversing).

Repeating for $p = 5$ we obtain:

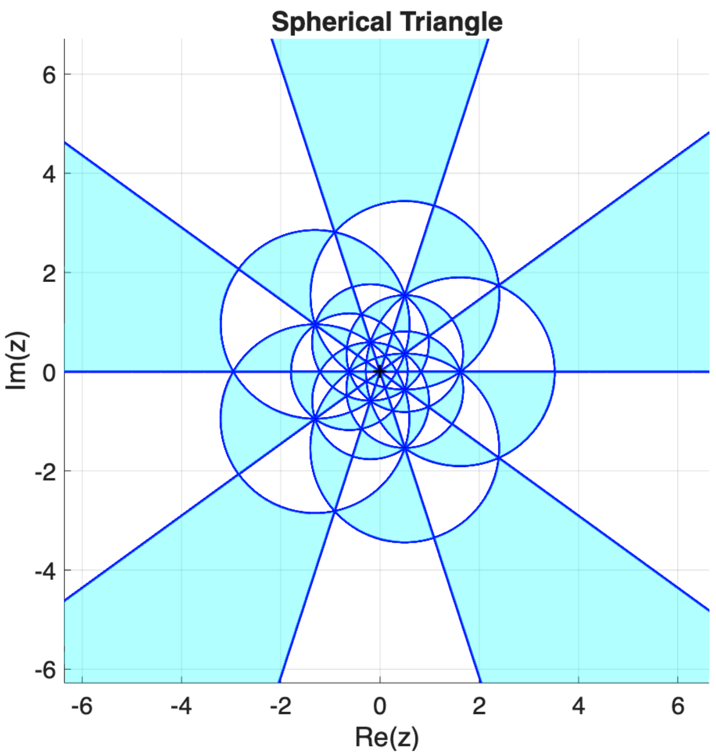

Figure 11: Admissible triangles for $p = 5$

We now get 60 non-overlapping triangles, and by the same argument as above for $p = 3$, we cannot obtain any new triangles by applying the S_i.

Finally for $p = 7$ we get:

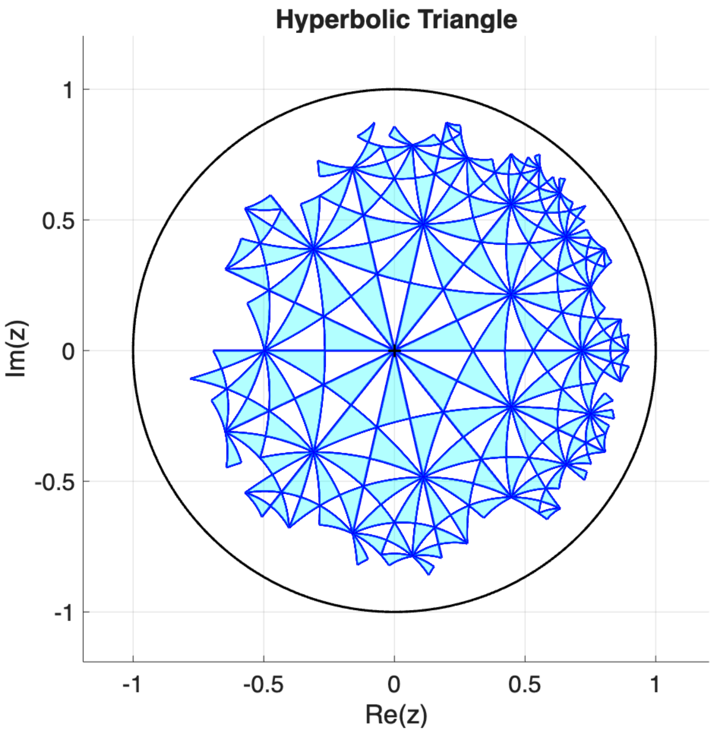

Figure 12: Admissible triangles for $p = 7$

We now obtain 168 non-overlapping triangles. However, unlike the spherical case we can indeed obtain new triangles by applying the S_i, this is reflected in the figure above as we do not have a tessellation of the disc model of the hyperbolic plane. This occurs since the hyperbolic plane is a non-compact surface (with unbounded lengths and regions), so we cannot obtain a tessellation with only finitely many triangles (as they would have finite area).

Question 6

The plot for $p = 3$ illustrates the orbit of Δ_0 under the action of the orientation preserving isometries of the tetrahedron (viewed as being inscribed on the sphere). As a result, each triangle corresponds to a unique (orientation preserving) isometry of the tetrahedron. We also get that S_1, S_2, S_3 are generators for the the group of orientation preserving isometries of the tetrahedron, which in turn is isomorphic to $\mathrm{PSL}(2, 3)$.

Likewise for $p = 5$, the plot shows the orbit of Δ_0 under the action of the orientation preserving isometries of the dodecahedron, and each triangle corresponds to a unique such isometry. This again yields an isomorphism from $\mathrm{PSL}(2, 5)$ to the group of orientation preserving isometries of the dodecahedron.

Since the group of orientation preserving isometries of the tetrahedron acts on its faces faithfully, we get a homomorphism ϕ to S_4. Since the isometries are orientation preserving, this has image in A_4, and comparing orders we get $\mathrm{PSL}(2, 3) \cong A_4$. We can also see this in Figure 10, by considering the 3 triangles that correspond to a face of the tetrahedron, and then noting that the transformations generated by the S_i are even permutations on this set.

You have $\mathrm{PSL}(2, 5) \cong A_5$. We can see this from our picture as follows: we can partition the vertices of the dodecahedron into 5 sets each of 4 equidistant vertices (and consider the corresponding set of triangles in Figure 11). We get 5 tetrahedra inscribed inside the dodecahedron, and since each orientation preserving isometry of the dodecahedron is an even permutation on these tetrahedra (since vertices are equidistant and they preserve orientation) we get the required isomorphism to A_5.

Programs for question 1

(i) stereo.m, invstereo.m, edge_plotter.m

```
function P = invstereo(z)
% Computes the inverse stereographic projection
    if isvector(z)
        z = z(:);
    end

    modz_sq = abs(z).^2;
    denom = modz_sq + 1;

    x = 2 * real(z) ./ denom;
    y = 2 * imag(z) ./ denom;
    z_coord = (modz_sq - 1) ./ denom;

    P = [x, y, z_coord];
end

function z = stereo(P)
% Stereographically projects a matrix of N points on the sphere
    x = P(:,1);
    y = P(:,2);
    z_coord = P(:,3);

    denom = 1 - z_coord;
    z = x ./ denom + 1i*(y ./ denom);
end

function edge = edge_plotter(p1,p2,n_points,tol)
% Calculates set of edge points for spherical arc between p1 and p2 and
% plots it
% Plots straight line if points are collinear with the origin
if abs(p1*conj(p2)-p2*conj(p1))< tol
    plot([real(p1),real(p2)],[imag(p1),imag(p2)],'b-','LineWidth',1.5);
    edge=linspace(p1,p2,n_points).';
% Calculates centre first then circular arc otherwise
else
    a=(p2*(abs(p1)^2-1)-p1*(abs(p2)^2-1))/(conj(p1)*p2-p1*conj(p2));
    rad_sq=abs(a)^2+1;
    r=sqrt(rad_sq);
    theta1=angle(p1-a);
    theta2=angle(p2-a);
    delta=theta2-theta1;
% Ensures we take the smaller arc by comparing lengths on the sphere
    delta=mod(delta+pi,2*pi)-pi;
    theta_arc1=theta1+linspace(0, delta, n_points);
    theta_arc2=theta1-linspace(0,sign(delta)*(2*pi - abs(delta)),n_points);
    arc_points1 = a + r * exp(1i*theta_arc1);
    arc_points2 = a + r * exp(1i*theta_arc2);

    arc_sphere1=invstereo(arc_points1);
    arc_sphere2=invstereo(arc_points2);
    arc_length1=zeros(1,n_points);
    arc_length2=zeros(1,n_points);

for m=1:n_points-1
    arc_length1(m)=acos(dot(arc_sphere1(m,:),arc_sphere1(m+1,:)));
    arc_length2(m)=acos(dot(arc_sphere2(m,:),arc_sphere2(m+1,:)));
end
```

```
l1=sum(arc_length1);
l2=sum(arc_length2);
if l1<l2
    edge = arc_points1.';
else
    edge = arc_points2.';
end
 % Plot the calculated shorter arc
    plot(real(edge), imag(edge), 'b-', 'LineWidth', 1.5);
end
end
```

(ii) plot_line.m

```
function edge =plot_line(p,n_points,sign)
% Plots line between point p and Inf through origin, with given orientation
if sign==1
    line=linspace(p,100*p,n_points);
    plot(real(line),imag(line),'b-','LineWidth',1.5);
    edge=line.';
elseif sign==-1
    line=linspace(100*p,p,n_points);
    plot(real(line),imag(line),'b-','LineWidth',1.5);
    edge=line.';
end
```

(iii) drawsphericaltriangle.m

```
function drawsphericaltriangle(z1,z2,z3)
% Draws and fills the spherical triangle in complex plane between z1, z2,
% z3 where these are all distinct and z1 can be Inf, in which case you have
% an unbounded region

% Number of points per triangle edge
n_points=1000;
tol=1e-9; % For edge cases
is_z1_inf=isinf(z1);

% Plots complex plane for set up
hold on; axis equal; grid on;
xlabel('Re(z)'); ylabel('Im(z)');
title('Spherical Triangle');
plot(0, 0, 'k+', 'MarkerSize', 8, 'LineWidth', 1.5);
set(gca,'FontSize',15)

% Set plot limits to ensure triangle is in figure
if ~is_z1_inf
    plot_lim=2*max(abs([z1,z2,z3]));
else
    plot_lim=2*max(abs([z2,z3]));
end

xlim([-plot_lim,plot_lim]); ylim([-plot_lim,plot_lim]);

store_edges=cell(1,4);

% Edge z1z2
if is_z1_inf
    store_edges{1}=plot_line(z2,n_points,1);
    zend1=store_edges{1}(end);
else
    store_edges{1}=edge_plotter(z1,z2,n_points,tol);
```

```
end

% Edge z2z3
if ~is_z1_inf
    store_edges{2}=edge_plotter(z2,z3,n_points,tol);
else
    store_edges{2}=edge_plotter(z3,z2,n_points,tol);
end

% Edge z3z1
if is_z1_inf
    store_edges{3}=plot_line(z3,n_points,-1);
    zend2=store_edges{3}(1);
else
    store_edges{3}=edge_plotter(z3,z1,n_points,tol);
end

% Constructs and fills boundary of triangle, if z1 is inf then fills part
% of unbounded region by adding another line between the ends of the rays to
    get a quadrilateral

boundary_points=[store_edges{1};store_edges{2};store_edges{3}];
if ~is_z1_inf
    fill(real(boundary_points),imag(boundary_points),'c','FaceAlpha',0.3,'
       EdgeColor','none')
else
    store_edges{4}=linspace(zend1,zend2,n_points).';
    boundary_points= [store_edges{1};store_edges{4};store_edges{3};
       store_edges{2}]; % Ensures correct orientation when region is filled
    fill(real(boundary_points),imag(boundary_points),'c','FaceAlpha',0.3,'
       EdgeColor','none')

end
```

Programs for question 2

(i) hypedge_plotter.m

```
function edge = hypedge_plotter(p1,p2,n_points,tol)
% Calculates set of edge points for hyperbolic arc between p1 and p2 and
% plots it
% Plots straight line if points are collinear with the origin
if abs(p1*conj(p2)-p2*conj(p1))< tol
    plot([real(p1),real(p2)],[imag(p1),imag(p2)],'b-','LineWidth',1.5);
    edge=linspace(p1,p2,n_points).';
% Calculates centre first then circular arc otherwise
else
    a=(p2*(abs(p1)^2+1)-p1*(abs(p2)^2+1))/(conj(p1)*p2-p1*conj(p2));
    rad_sq=abs(a)^2-1;
    r=sqrt(rad_sq);
    theta1=angle(p1-a);
    theta2=angle(p2-a);
    delta=theta2-theta1;
% Ensures we take the smaller arc
    delta=mod(delta+pi,2*pi)-pi;

    theta_arc=theta1+linspace(0, delta, n_points);
    arc_points = a + r * exp(1i * theta_arc);
    edge = arc_points.';
 % Plot the calculated shorter arc
    plot(real(arc_points), imag(arc_points), 'b-', 'LineWidth', 1.5);
```

```
end
```

(ii) drawhyperbolictriangle.m

```
function drawhyperbolictriangle(z1,z2,z3)
% Plots the hyperbolic triangle between z1,z2,z3 inside unit disk
% Number of points per triangle edge
n_points=1000;
tol=1e-9; % For edge cases
% Plots complex plane for set up
hold on; axis equal; grid on;
xlabel('Re(z)'); ylabel('Im(z)');
title('Hyperbolic Triangle');
plot(0, 0, 'k+', 'MarkerSize', 8, 'LineWidth', 1.5);
set(gca,'FontSize', 15);
xlim([-1.5,1.5]); ylim([-1.5,1.5]);
theta_uniform=linspace(-pi,pi,n_points);
unit_circle=exp(1i*theta_uniform);
plot(real(unit_circle),imag(unit_circle),'k','LineWidth',1.5);

store_edges=cell(1,3);

% Plots edges
store_edges{1}=hypedge_plotter(z1,z2,n_points,tol);
store_edges{2}=hypedge_plotter(z2,z3,n_points,tol);
store_edges{3}=hypedge_plotter(z3,z1,n_points,tol);

% Fills region in between
boundary_points=[store_edges{1};store_edges{2};store_edges{3}];
fill(real(boundary_points),imag(boundary_points),'c','FaceAlpha',0.3,'
    EdgeColor','none')
```

Programs for question 3

(i) q3sphericalsubdiv.m

```
% Script to plot the required spherical subdivisions for n=3,4,5
% Substitute in expressions for magnitude of vertices
r1=sqrt((sqrt(3)*tan(pi/3)-1)/(sqrt(3)*tan(pi/3)+1));
r2=sqrt((sqrt(3)*tan(pi/4)-1)/(sqrt(3)*tan(pi/4)+1));
r3=sqrt((sqrt(3)*tan(pi/5)-1)/(sqrt(3)*tan(pi/5)+1));
w1=sqrt((2*sin(pi/3)-1)/(2*sin(pi/3)+1));
w2=sqrt((2*sin(pi/4)-1)/(2*sin(pi/4)+1));
w3=sqrt((2*sin(pi/5)-1)/(2*sin(pi/5)+1));

% Plots subdivisions in 3 different figures
fig1=figure;
hold on;
for k=1:3
  drawsphericaltriangle(0,r1*exp(2*1i*(k-1)*pi/3),w1*exp(1i*(2*k-1)*pi/3));
  drawsphericaltriangle(0,w1*exp(1i*(2*k-1)*pi/3),r1*exp(2*1i*(k)*pi/3));
end
fig2=figure;
hold on;
for k=1:4
  drawsphericaltriangle(0,r2*exp(2*1i*(k-1)*pi/4),w2*exp(1i*(2*k-1)*pi/4));
  drawsphericaltriangle(0,w2*exp(1i*(2*k-1)*pi/4),r2*exp(2*1i*(k)*pi/4));
end
fig3=figure;
hold on;
for k=1:5
  drawsphericaltriangle(0,r3*exp(2*1i*(k-1)*pi/5),w3*exp(1i*(2*k-1)*pi/5));
  drawsphericaltriangle(0,w3*exp(1i*(2*k-1)*pi/5),r3*exp(2*1i*(k)*pi/5));
end
```

(ii) `q3hyperbolicsubdiv.m`

```
% Script to plot the required hyperbolic subdivisions for n=8,9
% Substitute in expressions for magnitude of vertices
r1=sqrt(-(sqrt(3)*tan(pi/7)-1)/(sqrt(3)*tan(pi/7)+1));
r2=sqrt(-(sqrt(3)*tan(pi/8)-1)/(sqrt(3)*tan(pi/8)+1));
w1=sqrt(-(2*sin(pi/7)-1)/(2*sin(pi/7)+1));
w2=sqrt(-(2*sin(pi/8)-1)/(2*sin(pi/8)+1));
% Plots subdivisions in 2 different figures
fig1=figure;
hold on;
for k=1:7
  drawhyperbolictriangle(0,r1*exp(2*1i*(k-1)*pi/7),w1*exp(1i*(2*k-1)*pi/7));
  drawhyperbolictriangle(0,w1*exp(1i*(2*k-1)*pi/7),r1*exp(2*1i*(k)*pi/7));
end
fig2=figure;
hold on;
for k=1:8
  drawhyperbolictriangle(0,r2*exp(2*1i*(k-1)*pi/8),w2*exp(1i*(2*k-1)*pi/8));
  drawhyperbolictriangle(0,w2*exp(1i*(2*k-1)*pi/8),r2*exp(2*1i*(k)*pi/8));
end
```

Programs for question 5

(i) `reflection.m, hyperbolic_reflection.m`

```
function transformed_z = reflection(p1,p2,z)
% Reflects z in the spherical line through p1, p2
tol=1e-9;
large_mod=1e9;
% Handle inf
 if abs(p1)>large_mod
     p1=inf;
 end
 if abs(p2)>large_mod
     p2=inf;
 end
 if abs(z)> large_mod
     z=inf;
 end

 % Apply euclidean reflection or inversion depending on which case
 if ~isinf(z)

 if abs(p1*conj(p2)-p2*conj(p1))< tol
    theta=angle(p1-p2);
    transformed_z=exp(2i*theta)*conj(z);
 elseif p1 == inf
    theta=angle(p2);
    transformed_z=exp(2i*theta)*conj(z);
 elseif p2 == inf
     theta=angle(p1);
     transformed_z=exp(2i*theta)*conj(z);
 else
     a=sphericalcentre(p1,p2);
     if abs(z-a)<tol
         transformed_z=inf;
     else
        transformed_z=(a*conj(z)+1)/(conj(z-a));
     end
 end

 else
     if abs(p1*conj(p2)-p2*conj(p1))< tol
         transformed_z=inf;
     else
         a=sphericalcentre(p1,p2);
         transformed_z=a;
     end
 end

function transformed_z = hyperbolic_reflection(p1,p2,z)
tol=1e-9;
if abs(p1*conj(p2)-p2*conj(p1))< tol
    theta=angle(p1-p2);
    transformed_z=exp(2i*theta)*conj(z);
else
    a=(p2*(abs(p1)^2+1)-p1*(abs(p2)^2+1))/(conj(p1)*p2-p1*conj(p2));
     transformed_z=(a*conj(z)-1)/(conj(z-a));
end
```

(iii) S1.m, S2.m, S3.m

```
function array = S1(z1,z2,z3)
temp3=reflection(z1,z2,z3);
array=[z1,reflection(z1,z3,z2),reflection(z1,z3,temp3)];

function array = S2(z1,z2,z3)
temp1=reflection(z2,z3,z1);
w1=reflection(z1,z2,temp1);
w2=z2;
w3=reflection(z1,z2,z3);
array=[w1,w2,w3];

function array = S3(z1,z2,z3)
w1=reflection(z2,z3,z1);
temp2=reflection(z1,z3,z2);
w2=reflection(z2,z3,temp2);
w3=z3;
array=[w1,w2,w3];
```

(iv) S1hyp.m, S2hyp.m, S3hyp.m

```
function array = S1hyp(z1,z2,z3)
temp3=hyperbolic_reflection(z1,z2,z3);
array=[z1,hyperbolic_reflection(z1,z3,z2),hyperbolic_reflection(z1,z3,temp3)
   ];

function array = S2hyp(z1,z2,z3)
temp1=hyperbolic_reflection(z2,z3,z1);
w1=hyperbolic_reflection(z1,z2,temp1);
w2=z2;
w3=hyperbolic_reflection(z1,z2,z3);
array=[w1,w2,w3];

function array = S3hyp(z1,z2,z3)
w1=hyperbolic_reflection(z2,z3,z1);
temp2=hyperbolic_reflection(z1,z3,z2);
w2=hyperbolic_reflection(z2,z3,temp2);
w3=z3;
array=[w1,w2,w3];
```

(v) reorder.m, drawsphericaltriangleq5.m

```
function reordered_triple = reorder(z1, z2, z3)
% Reorders a triple with Inf to the first position.
% Takes three complex numbers (or Inf) as separate inputs.

    triple = [z1, z2, z3]; % Create a row vector

    if isinf(triple(1)) || abs(triple(1))>1e7
        reordered_triple = [inf,triple(2),triple(3)];
    elseif isinf(triple(2)) || abs(triple(2))>1e7
        reordered_triple = [inf, triple(1), triple(3)];
    elseif isinf(triple(3)) || abs(triple(3))>1e7
        reordered_triple = [inf, triple(1), triple(2)];
    else
        reordered_triple = triple;
    end
end

% General variant of drawsphericaltriangle for question 5
function drawsphericaltriangleq5(z1,z2,z3)
triple=reorder(z1,z2,z3);
drawsphericaltriangle(triple(1),triple(2),triple(3))
```

(vi) lex_less.m, canonical_psl_matrix.m

```
 function flag = lex_less(v1, v2)
% Returns true if vector v1 is lexicographically less than v2
    flag = false;
    for idx = 1:length(v1)
        if v1(idx) < v2(idx)
            flag = true;
            return;
        elseif v1(idx) > v2(idx)
            flag = false;
            return;
        end
    end
end

function M_can = canonical_psl_matrix(M,p)
% Returns the canonical representative of the matrix M in PSL(2,p)
% Out of M and -M (mod p), we choose the one with a lexicographically
   smaller vector
    M_mod = mod(M, p);
    M_neg = mod(-M, p);
    vec1 = M_mod(:);
    vec2 = M_neg(:);
    if lex_less(vec1, vec2)
        M_can = M_mod;
    else
        M_can = M_neg;
    end
end
```

(vii) q5sphericalplot_3.m, q5sphericalplot_5.m, q5hyperbolicplot_7.m

```
% Script to compute all triples of admissible vertices in the case that p=3
% We compute the corresponding PSL(2,3) elements as products of generators
% then use this to generate the new triple of vertices
sigma1=[0 1; -1 0];
sigma2= [0 -1; 1 -1];
```

```matlab
sigma3= [1 1; 0 1];
generators={sigma1,sigma2,sigma3};
tol=1e-9;

r=sqrt((sqrt(3)*tan(pi/3)-1)/(sqrt(3)*tan(pi/3)+1));
w=sqrt((2*sin(pi/3)-1)/(2*sin(pi/3)+1))*exp(1i*pi/3);

initial_triple=[w,r,0];
S1=@(triple) S1(triple(1),triple(2),triple(3));
S2=@(triple) S2(triple(1),triple(2),triple(3));
S3=@(triple) S3(triple(1),triple(2),triple(3));
transform_func={S1,S2,S3};

% Since PSL(2,3) has 12 elements (Note we can also run this without a
% target count but will take longer)
target_count=12;


matrices = {canonical_psl_matrix(eye(2),3)};
triples  = [initial_triple];

% Start generating matrices and triples
while length(matrices) < target_count
    n = length(matrices);
    new_found = false;

    for l = 1:n
        curr_matrix = matrices{l};
        curr_triple = triples(l,:);

        for k = 1:3
            next_matrix = mod(curr_matrix*generators{k}, 3);
            canon_next  = canonical_psl_matrix(next_matrix,3);
            already_found = any(cellfun(@(A) isequal(A, canon_next),
               matrices));
            next_triple = transform_func{k}(curr_triple);

            if ~already_found
                matrices{end+1} = canon_next;
                triples(end+1, :) = next_triple;
                new_found = true;

                if length(matrices) >= target_count
                    break;
                end
            end
        end
        if length(matrices) >= target_count
            break;
        end
    end

    if ~new_found
        break;
    end
end
% Plots triangles
figure;
hold on

for k=1:12
```

```
    x=triples(k,:);
    drawsphericaltriangleq5(x(1),x(2),x(3))
end

% Script to compute all triples of admissible vertices in the case that p=5
% We compute the corresponding PSL(2,5) elements as products of generators
% then use this to generate the new triple of vertices
sigma1=[0 1; -1 0];
sigma2= [0 -1; 1 -1];
sigma3= [1 1; 0 1];
generators={sigma1,sigma2,sigma3};
tol=1e-9;

r=sqrt((sqrt(3)*tan(pi/5)-1)/(sqrt(3)*tan(pi/5)+1));
w=sqrt((2*sin(pi/5)-1)/(2*sin(pi/5)+1))*exp(1i*pi/5);

initial_triple=[w,r,0];
S1=@(triple) S1(triple(1),triple(2),triple(3));
S2=@(triple) S2(triple(1),triple(2),triple(3));
S3=@(triple) S3(triple(1),triple(2),triple(3));
transform_func={S1,S2,S3};

% Since PSL(2,5) has 60 elements (Note we can also run this without a
% target count but will take longer)
target_count=60;


matrices = {canonical_psl_matrix(eye(2),5)};
triples  = [initial_triple];

% Start generating matrices and triples
while length(matrices) < target_count
    n = length(matrices);
    new_found = false;

    for l = 1:n
        curr_matrix = matrices{l};
        curr_triple = triples(l,:);

        for k = 1:3
            next_matrix = mod(curr_matrix*generators{k}, 5);
            canon_next  = canonical_psl_matrix(next_matrix,5);
            already_found = any(cellfun(@(A) isequal(A, canon_next),
               matrices));
            next_triple = transform_func{k}(curr_triple);

            if ~already_found
                matrices{end+1} = canon_next;
                triples(end+1, :) = next_triple;
                new_found = true;

                if length(matrices) >= target_count
                    break;
                end
            end
        end
        if length(matrices) >= target_count
            break;
        end
    end
    if ~new_found
        break;
    end
end
% Plots triangles
figure;
hold on

for k=1:target_count
    x=triples(k,:);
    drawsphericaltriangleq5(x(1),x(2),x(3))
end

% Script to compute all triples of admissible vertices in the case that p=7
% We compute the corresponding PSL(2,7) elements as products of generators
% then use this to generate the new triple of vertices
sigma1=[0 1; -1 0];
sigma2= [0 -1; 1 -1];
sigma3= [1 1; 0 1];
generators={sigma1,sigma2,sigma3};
tol=1e-9;

r=sqrt(-(sqrt(3)*tan(pi/7)-1)/(sqrt(3)*tan(pi/7)+1));
w=sqrt(-(2*sin(pi/7)-1)/(2*sin(pi/7)+1))*exp(1i*pi/7);

initial_triple=[w,r,0];
S1hyp=@(triple) S1hyp(triple(1),triple(2),triple(3));
S2hyp=@(triple) S2hyp(triple(1),triple(2),triple(3));
S3hyp=@(triple) S3hyp(triple(1),triple(2),triple(3));
transform_func={S1hyp,S2hyp,S3hyp};

% Since PSL(2,7) has 168 elements (Note we can also run this without a
% target count but will take longer)
target_count=168;


matrices = {canonical_psl_matrix(eye(2),7)};
triples  = [initial_triple];

% Start generating matrices and triples
while length(matrices) < target_count
    n = length(matrices);
    new_found = false;

    for l = 1:n
        curr_matrix = matrices{l};
        curr_triple = triples(l,:);

        for k = 1:3
            next_matrix = mod(curr_matrix*generators{k}, 7);
            canon_next  = canonical_psl_matrix(next_matrix,7);
            already_found = any(cellfun(@(A) isequal(A, canon_next),
               matrices));
            next_triple = transform_func{k}(curr_triple);

            if ~already_found
                matrices{end+1} = canon_next;
                triples(end+1, :) = next_triple;
                new_found = true;

                if length(matrices) >= target_count
```

```
                    break;
                end
            end
        end
        if length(matrices) >= target_count
            break;
        end
    end

    if ~new_found
        break;
    end
end
% Plots triangles
figure;
hold on

for k=1:target_count
    x=triples(k,:);
    drawhyperbolictriangle(x(1),x(2),x(3))
end
```