Arithmetic of Elliptic Curves

Galen Ballew & James Duncan

May 6, 2016

Abstract

Our research focuses on 9 specific elliptic curves E over $\mathbb{Q}$, each with complex multiplication by the maximal order in an imaginary quadratic field. Viewed over $\mathbb{C}$, each E gives rise to tori, defined by the generators $\omega_1, \omega_2 \in \mathbb{C}$ of the period lattice. Using SAGE, information and characteristics about the curves and their tori were calculated and compiled. Additionally, the tori were virtually constructed using 3D modeling software. These virtual meshes were converted to G-Code and printed on an Ultimaker2 3D printer.

1 Motivation

Elliptic curves are interesting mathematical phenomena. Certain curves can be used to solve Diophantine equations (for example, in the proof of Fermat's Last Theorem), part of factoring algorithms, or used in cryptography. This research project is about developing an understanding of elliptic curves, their properties, and creating visualizations of them.

2 Elliptic Curves over $\mathbb{Q}$

In order to define an elliptic curve, we have to consider the more general class of curves to which elliptic curves belong.

Definition 1. A **cubic plane curve** C defined over a field K is a curve in the plane given by the equation

$$y^2 = x^3 + ax + b$$

for $a, b \in K$. The **discriminant** of C is defined as

$$\Delta = -16(4a^3 + 27b^2).$$

Theorem 1. *Let C be a cubic plane curve. Then C is non-singular if and only if its discriminant is non-zero.*

Proof. ($\Rightarrow$) Let $f(x) = x^3 + ax + b$. Define $F(x, y) = y^2 - x^3 - ax - b$, the left-hand side of the above equation. Then points (x', y') such that $F(x', y') = 0$ correspond to points on the cubic curve.

Let C be non-singular. Then there are no points on the curve where both partial derivatives of F vanish simultaneously. Suppose (x_0, y_0) is a point such that $\frac{\partial F}{\partial x} = \frac{\partial F}{\partial y} = 0$:

$$\frac{\partial F}{\partial x}(x_0) = -3x_0^2 - a = 0,$$

$$\frac{\partial F}{\partial y}(y_0) = 2y_0 = 0.$$

Then $y_0 = 0$ and $x_0 = \pm\sqrt{-a/3}$. Note that since $f'(x_0) = 0$ and $f(x_0) = 0$, x_0 is a double root of f. We see that in this case, $\Delta = 0$. Without loss of generality, let $x_0 = \sqrt{-a/3}$. Then setting $F(x_0, y_0) = 0$, we have:

$$\begin{aligned} 0 &= -\left(\sqrt{\frac{-a}{3}}\right)^3 - a\sqrt{\frac{-a}{3}} - b \\ &= \frac{a}{3}\sqrt{\frac{-a}{3}} - a\sqrt{\frac{-a}{3}} - b \\ &= \frac{-2a}{3}\sqrt{\frac{-a}{3}} - b. \end{aligned}$$

It follows that

$$\begin{aligned} b^2 &= \frac{4a^2}{9}\left(\frac{-a}{3}\right) \\ &= \frac{-4a^3}{27}. \end{aligned}$$

So $4a^3 + 27b^2 = 0$, which implies that the discriminant is zero.

The case of $x_0 = -\sqrt{-a/3}$ follows similarly.

$$\begin{aligned} b^2 &= \left(\frac{2a}{3}\sqrt{\frac{-a}{3}}\right)^2 \\ &= -\frac{4}{27}a^3. \end{aligned}$$

Again, this implies that the discriminant is zero.

($\Leftarrow$) Now let $\Delta \neq 0$. We will show that this implies that the partial derivatives of F will never vanish simultaneously, so there are no singular points on C. Note that $\Delta \neq 0$ implies

that the 3 roots of $f(x)$ are distinct. Thus, f and f' will not share a common root. That is, if $(x_0, 0)$ is on C then $\frac{\partial F}{\partial x}|_{x_0} \neq 0$. In other words, the partial derivatives will not simultaneously vanish. C is therefore non-singular. □

Definition 2. An **elliptic curve** E over the field K is a cubic curve in the projective plane with non-zero discriminant, defined by the **Weierstrass equation**

$$E_{a,b} : y^2 = x^3 + ax + b,$$

together with a fixed K-rational point $\mathcal{O} \in E$, called the **point at infinity**, given in projective coordinates as $[0 : 1 : 0]$.

For the rest of this section we will assume that $K = \mathbb{Q}$. As we will see, the rational points on $E/\mathbb{Q}$, which we denote $E(\mathbb{Q})$, form an abelian group. The group law is defined as follows.

Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be distinct points on E. Define $P * Q = (x_3, y_3)$ as the third point of intersection with E on the secant line through P and Q. Define $P + Q$ as $(P * Q) * \mathcal{O}$. The secant line through any point on the curve and the point at infinity is a vertical line, so because the curve is symmetric about the x-axis, $P + Q = (x_3, -y_3)$.

The line through P and Q has the equation

$$y = \lambda x + \nu, \text{ where } \lambda = \frac{y_2 - y_1}{x_2 - x_1} \text{ and } \nu = y_1 - \lambda x_1 = y_2 - \lambda x_2.$$

To find a formula for $P * Q$, we substitute $y = \lambda x + \nu$ into the equation for E, giving

$$(\lambda x + \nu)^2 = x^3 + ax + b.$$

Moving everything to one side gives

$$0 = x^3 - \lambda^2 x^2 + (a - 2\lambda\nu)x + (b - \nu^2). \tag{1}$$

This is a cubic equation in x which has three roots, so we can rewrite the equation as

$$0 = (x - x_1)(x - x_2)(x - x_3) \tag{2}$$

since we know that the roots of (1) occur exactly where the line $y = \lambda x + \nu$ meets the curve.

Matching the coefficients of the x^2 term of (1) and (2) gives

$$\lambda^2 = x_1 + x_2 + x_3$$

so $x_3 = \lambda^2 - x_1 - x_2$. Thus $y_3 = \lambda x_3 + \nu$. Finally, $P + Q = (x_3, -y_3)$.

However, if we want to add a point $P = (x_1, y_1)$ on E to itself, these formulas will not work because the denominator of λ will be zero. In that case, we can implicitly differentiate $y^2 = x^3 + ax + b$ giving $2ydy = (3x^2 + a)dx$ and then the slope of the tangent line at P is

$$\lambda = \frac{dy}{dx} = \frac{3x_1^2 + a}{2y_1}$$

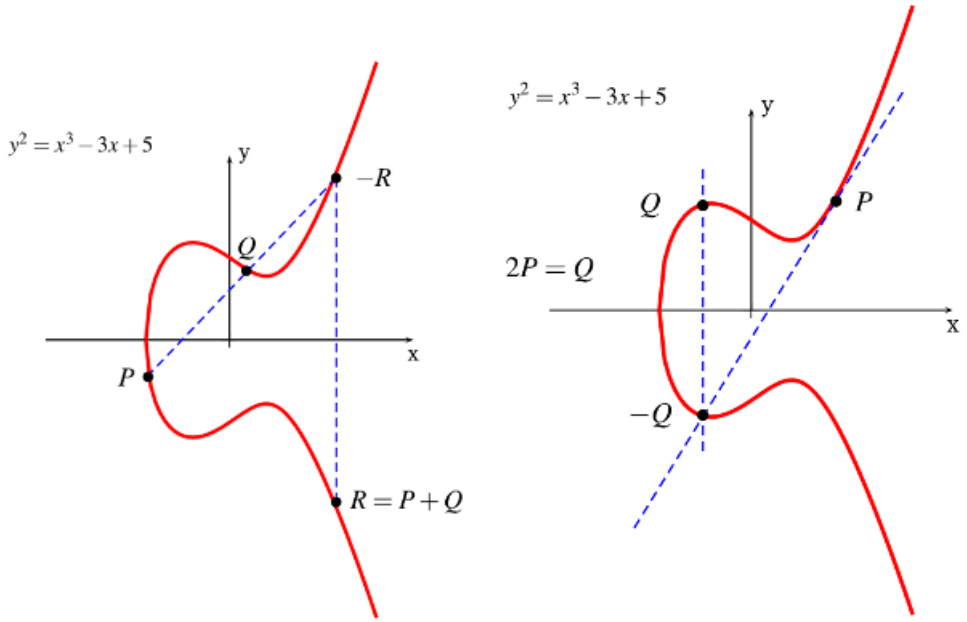

Figure 1: Adding points on the curve.

which we can then use to find $P + P = 2P$ just as before.

Figure 1 shows a visualization of the group law. This operation of addition is commutative: the line through P and Q is the same as that through Q and P. Moreover, the point $\mathcal{O}$ is the identity element, because for any $P \neq \mathcal{O}$, the line through P and $\mathcal{O}$ is a vertical line, and thus the third point of intersection of that line with the curve is $-P$. Reflecting across the x-axis gives P again. The line connecting $\mathcal{O}$ to itself is the line at infinity, which intersects with E at $\mathcal{O}$. And every point P also has an inverse, namely $-P$, by the symmetry of the curve. Proving the associativity of the group law would take many pages of calculations, but can be done using the explicit addition formulas given above.

We present the following theorem without proof.

Theorem 2. Mordell's Theorem
Let $E/\mathbb{Q}$ be an elliptic curve, and let $E(\mathbb{Q})$ be the group of points on $E/\mathbb{Q}$. Then

$$E(\mathbb{Q}) \simeq \mathbb{Z}^r \oplus E(\mathbb{Q})_{tors}$$

*where $r = r(E)$ is some non-negative integer, called the **arithmetic rank** of $E/\mathbb{Q}$, and where $E(\mathbb{Q})_{tors}$ is the group of points of finite order in $E(\mathbb{Q})$, called the **torsion subgroup** of $E(\mathbb{Q})$.*

This important result, proved by Louis Mordell in 1922, tells us that $E(\mathbb{Q})$ is a finitely generated abelian group. That is, given a finite number of rational points on $E/\mathbb{Q}$ we can arrive at all other points in $E(\mathbb{Q})$ using the group addition law.

3 Complex Points on $E/\mathbb{C}$

We have thus far considered the rational points $E(\mathbb{Q})$ on an elliptic curve E defined over the rationals. We now shift our focus to elliptic curves with complex coefficients, $E/\mathbb{C}$. The group law described above was purely algebraic, and so applies equally well to $E(\mathbb{R})$, the points with real coordinates satisfying the Weierstrass equation $E_{a,b}$, and $E(\mathbb{C})$, the points with complex coordinates. Moreover, if we assume $a, b \in \mathbb{Q}$, these groups satisfy the following hierarchy:

$$\{\mathcal{O}\} \subset E(\mathbb{Q}) \subset E(\mathbb{R}) \subset E(\mathbb{C}).$$

However, we cannot visualize $E(\mathbb{C})$ in the same way that we can $E(\mathbb{R})$ (see Figure 1). Therefore, our goal in this section will be to move from the picture of $E(\mathbb{R})$ in $\mathbb{R}^2$ that we have already seen to a picture of $E(\mathbb{C})$ as an embedding of a torus in the complex projective plane and 3D visualizations of that torus.

To do so, we turn to the theory of elliptic functions. By replacing x and y by $4x$ and $4y$ in the Weierstrass equation, we get

$$y^2 = 4x^3 - g_2 x - g_3. \tag{3}$$

If g_2 and g_3 are such that the polynomial on the right has distinct roots, which is the case when $\Delta \neq 0$, then we may find complex numbers ω_1 and ω_2 called **periods**. We may find approximations of ω_1 and ω_2 to arbitrary precision using the following method.

To simplify things, let us assume that our curve is defined over $\mathbb{R}$. Let a and b be positive real numbers, and define a_n and b_n by

$$\begin{aligned} a_0 &= a,\ b_0 = b, \\ a_n &= \frac{1}{2}(a_{n-1} + b_{n-1}), \\ b_n &= \sqrt{a_{n-1} b_{n-1}}. \end{aligned}$$

a_n is the arithmetic mean (average) of the previous values a_{n-1} and b_{n-1}, while b_n is their geometric mean. These number converge quickly to a shared value.

Proposition 1. *Suppose $a \geq b > 0$. Then, for all n*

$$b_{n-1} \leq b_n \leq a_n \leq a_{n-1}$$

and

$$0 \leq a_n - b_n \leq \frac{1}{2}(a_{n-1} - b_{n-1}).$$

Therefore,

$$M(a, b) = \lim_{n \to \infty} a_n = \lim_{n \to \infty} b_n$$

exists.

Proof. If we assume that $a_{n-1} \geq b_{n-1}$, the quantity $\sqrt{a_{n-1}} - \sqrt{b_{n-1}}$ is non-negative. Using this quantity, We have

$$a_n - b_n = \frac{1}{2}(\sqrt{a_{n-1}} - \sqrt{b_{n-1}})^2 \geq 0,$$

so we see that $b_n \leq a_n$. Further, we can see that

$$b_{n-1} = \sqrt{b_{n-1}b_{n-1}} \leq \sqrt{a_{n-1}b_{n-1}} = b_n$$

and

$$a_n = \frac{1}{2}(a_{n-1} + b_{n-1}) \leq \frac{1}{2}(a_{n-1} + a_{n-1}) = a_{n-1}.$$

Moreover,

$$\begin{aligned} a_n - b_n &= \frac{1}{2}(\sqrt{a_{n-1}} - \sqrt{b_{n-1}})^2 \\ &\leq \frac{1}{2}(\sqrt{a_{n-1}} - \sqrt{b_{n-1}})(\sqrt{a_{n-1}} + \sqrt{b_{n-1}}) \\ &= \frac{1}{2}(a_{n-1} - b_{n-1}). \end{aligned}$$

So, $a_n - b_n \leq (1/2)^n(a-b)$, and thus $a_n - b_n \to 0$. The limit $M(a,b)$ exists because the a_n are a decreasing sequence bounded below by the b_n, which themselves are increasing and converging to the same limit as the a_n's. □

$M(a,b)$ is known as the **arithmetic-geometric mean** of a and b. We can now express the periods ω_1 and ω_2 using such means.

Theorem 3. *Let E be given by*

$$y^2 = 4x^3 - g_2x - g_3 = 4(x-e_1)(x-e_2)(x-e_3).$$

First assume that $e_1 < e_2 < e_3$ are all real. Then the periods ω_1 and ω_2 are defined by

$$\begin{aligned} \omega_1 &= \frac{\pi i}{M(\sqrt{e_3 - e_1}, \sqrt{e_2 - e_1})}, \\ \omega_2 &= \frac{\pi}{M(\sqrt{e_3 - e_1}, \sqrt{e_3 - e_2})}. \end{aligned}$$

Now, suppose that e_1 is the unique real root of $4x^3 - g_2x - g_3$. Let $e' = \sqrt{3e_1^2 - (1/4)g_2}$. Then

$$\begin{aligned} \omega_1 &= \frac{2\pi}{M(\sqrt{4e'}, \sqrt{2e' - 3e_1})}, \\ \omega_2 &= -\frac{\omega_1}{2} + \frac{\pi i}{M(\sqrt{4e'}, \sqrt{2e' - 3e_1})}. \end{aligned}$$

The proof of the preceding theorem can be found in [7]. With the periods ω_1 and ω_2 in hand, we define a lattice, Λ, in the complex plane by

$$\Lambda = \mathbb{Z}\omega_1 + \mathbb{Z}\omega_2 = \{m\omega_1 + n\omega_2, m, n \in \mathbb{Z}\}.$$

Then, considering the complex plane mod the lattice, $\mathbb{C}/\Lambda$, gives us the **fundamental parallelogram**, Π. This parallelogram is the embedding of the torus associated to the curve in the complex plane.

There is a one-to-one correspondence between points on $E/\mathbb{C}$ and points in the fundamental parallelogram. Given the lattice Λ, the **Weierstrass $\wp$ -function** is defined as

$$\wp(u) = \frac{1}{u^2} + \sum_{\substack{\omega\in\Lambda \\ \omega\neq 0}} \left(\frac{1}{(u-\omega)^2} - \frac{1}{\omega^2} \right)$$

for all $u \in \mathbb{C}$. This function is doubly periodic, with periods ω_1 and ω_2, i.e.,

$$\wp(u+\omega) = \wp(u) \text{ for all } u \in \mathbb{C} \text{ and all } \omega \in \Lambda.$$

Furthermore, it can be shown that $\wp(u)$ satisfies

$$\wp'(u)^2 = 4\wp(u)^3 - g_2\wp(u) - g_3.$$

Notice that this is exactly the form of the Weierstrass equation that we gave in (3). Thus, $P(u) = (\wp(u), \wp'(u))$ is a point on $E/\mathbb{C}$. We have thus defined a map from points in $\mathbb{C}$ to $E/\mathbb{C}$.

This map is onto the curve. Restricting our attention to only those points on Π – while mapping the corners of that parallelogram to $\mathcal{O}$ and identifying the opposite sides – the map also becomes one-to-one.

Figure 2 shows how one might go from a parallelogram to a torus. There are many ways one could construct a torus from Π. In our case, we fold the fundamental parallelogram in two ways. In the first, the major radius R_1 of the torus is defined using ω_1 by $R_1 = |\omega_1|/2\pi$, where $|z|$ is the distance from the origin to z. That is, ω_1 defines the length of the tube shown in the third panel of Figure 2. The circumference of that tube is then defined by the height of Π, so that the minor radius is given by

$$r_1 = \frac{|\omega_2| \sin\theta}{2\pi},$$

where θ is the angle between ω_1 and ω_2. To produce a second torus from Π, we use $R_2 = |\omega_2|/2\pi$ and $r_2 = |\omega_1| \sin\theta/2\pi$.

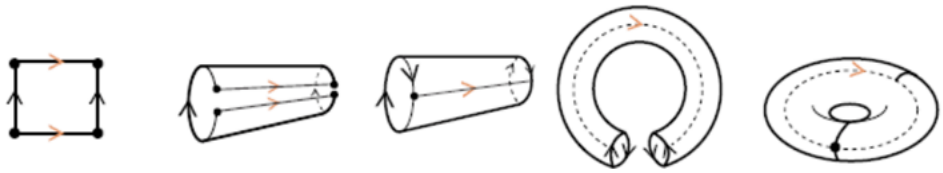

Figure 2: The parallelogram folds into the torus much as one would fold a sheet of paper.

4 $E/\mathbb{Q}$ with CM

Theorem 4. *Let $E/\mathbb{Q}$ be an elliptic curve and write the ring of all isogenies of $E/\overline{\mathbb{Q}}$ as*

$$\mathrm{End}_{\overline{\mathbb{Q}}}(E) := \{\varphi : E(\overline{\mathbb{Q}}) \to E(\overline{\mathbb{Q}}) \text{ group homomorphism}\}.$$

Then, either

$$\mathrm{End}_{\overline{\mathbb{Q}}}(E) \simeq \mathbb{Z},$$

*in which case we say that $E/\mathbb{Q}$ is **without Complex Multiplication** (non-CM), or*

$$\exists D \in \{1, 2, 3, 7, 11, 19, 43, 67, 163\}$$

and an order $\mathcal{O}$ in an imaginary quadratic field $\mathbb{Q}(\sqrt{-D})$ such that

$$\mathrm{End}_{\overline{\mathbb{Q}}}(E) \simeq \mathcal{O},$$

*in which case we say that $E/\mathbb{Q}$ is **with Complex Multiplication** (CM) and $\mathbb{Q}(\sqrt{-D})$ is its CM field.*

In our application, we're interested in the 9 elliptic curves with CM field $\mathbb{Q}(\sqrt{-D})$ and $\mathrm{End}(E) = \mathcal{O}_{\mathbb{Q}(\sqrt{-D})}$ for each $D \in \{1, 2, 3, 7, 11, 19, 43, 67, 163\}$. Relevant data for each curve is given in Table 1. The graphs of the curves are in 5. Included in the data are computations for the **conductor** and **j-invariant** (refer to Theorem 2 for discussions of torsion and rank.). Informally, the conductor of a curve E is an arithmetic invariant $N = N_E \in \mathbb{Z}$ that gives precise information about the reduction of E modulo each prime p. The j-invariant, meanwhile, defines isomorphism classes of elliptic curves.

Definition 3. Let E be an elliptic curve over $\mathbb{C}$ defined by $y^2 = 4x^3 - g_2x - g_3$ via the Weierstrass elliptic functions. Then the j-invariant is defined as

$$j = 1728\frac{g_2^3}{\Delta} = 1728\frac{g_2^3}{g_2^3 - 27g_3^2}.$$

Label	*Discriminant*	*Conductor*	*Torsion*	*Rank*	*CM Field*
64a4	-64	64	$\mathbb{Z}/2$	0	$\mathbb{Q}(\sqrt{-1})$
	j-invariant: 1728				
	$y^2 = x^3 + x$				
256a1	512	256	$\mathbb{Z}/2$	1	$\mathbb{Q}(\sqrt{-2})$
	j-invariant: 8000				
	$y^2 = x^3 + 4x^2 + 2x$				
27a3	-27	27	$\mathbb{Z}/3$	0	$\mathbb{Q}(\sqrt{-3})$
	j-invariant: 0				
	$y^2 + y = x^3$				
49a1	-343	49	$\mathbb{Z}/2$	0	$\mathbb{Q}(\sqrt{-7})$
	j-invariant: -3375				
	$y^2 + xy = x^3 - x^2 - 2x - 1$				
121b1	-1331	121	Trivial	1	$\mathbb{Q}(\sqrt{-11})$
	j-invariant: -32768				
	$y^2 + y = x^3 - x^2 - 7x + 10$				
361a1	-6859	361	Trivial	1	$\mathbb{Q}(\sqrt{-19})$
	j-invariant: -884736				
	$y^2 + y = x^3 - 38x + 90$				
1849a1	-79507	1849	Trivial	1	$\mathbb{Q}(\sqrt{-43})$
	j-invariant: -884736000				
	$y^2 + y = x^3 - 860x^2 + 9707$				
4489a1	-300763	4489	Trivial	1	$\mathbb{Q}(\sqrt{-67})$
	j-invariant: -14719795200				
	$y^2 + y = x^3 - 7370x^2 + 243528$				
26569a1	-4330747	26569	Trivial	1	$\mathbb{Q}(\sqrt{-163})$
	j-invariant: -262537412640768000				
	$y^2 + y = x^3 - 2174420x + 1234136692$				

Table 1

5 Application

SageMath was used throughout this project as a means to produce information about the 9 elliptic curves we were interested in. See the Appendix for our full code, which includes visualization of the curves and tori in Sage.

SageMath was used to calculate ω_1 and ω_2 for each of the period lattices defined from the curves by the method of arithmetic-geometric means discussed above. In turn, ω_1 and ω_2 were used to define major and minor radii for two distinct tori per curve. These tori were constructed in the 3D modeling software Autodesk Maya.

Similar to the wireframe renderings in Figure 3, the meshes of the tori were converted into G-code using the program Cura. G-code is a programming language for machine tools.

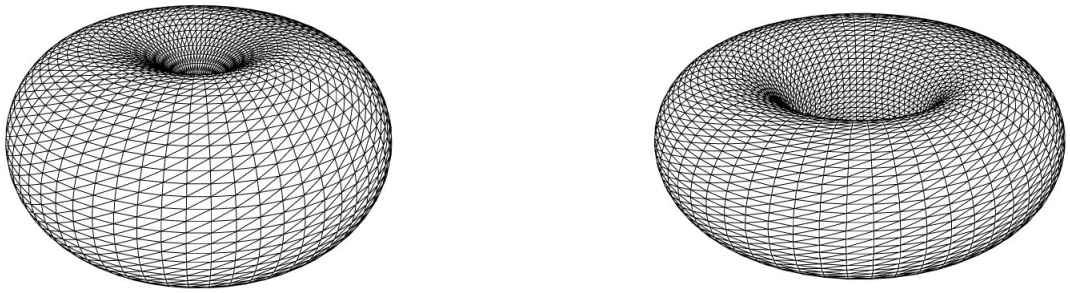

Figure 3: Tori generated by elliptic curve *64a4*, with major radii defined by ω_1 and ω_2 respectively.

It converts the 3D mesh into (X,Y,Z) coordinates for the printhead of the Ultimaker2. Figure 6 shows the 3D meshes for each torus.

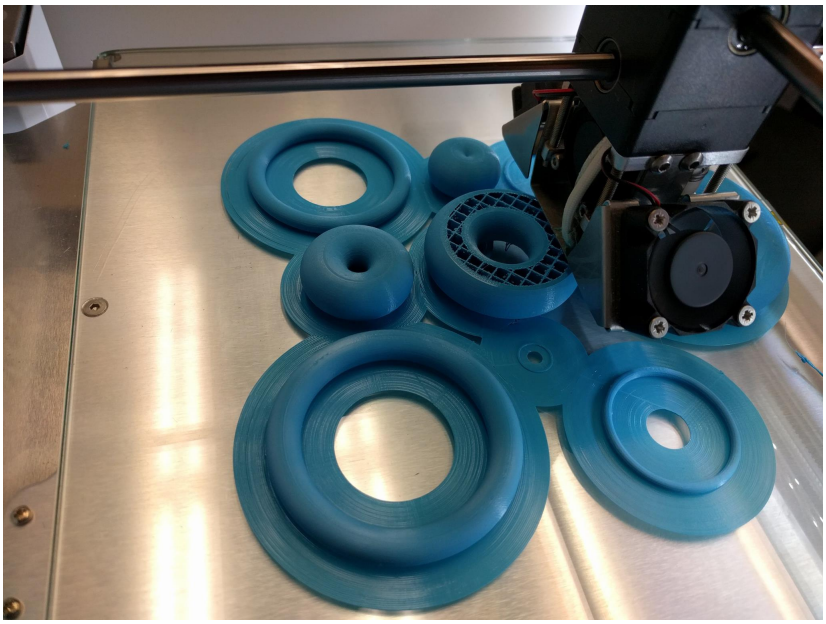

Figure 4: Nine of the tori in a batch 3D print job.

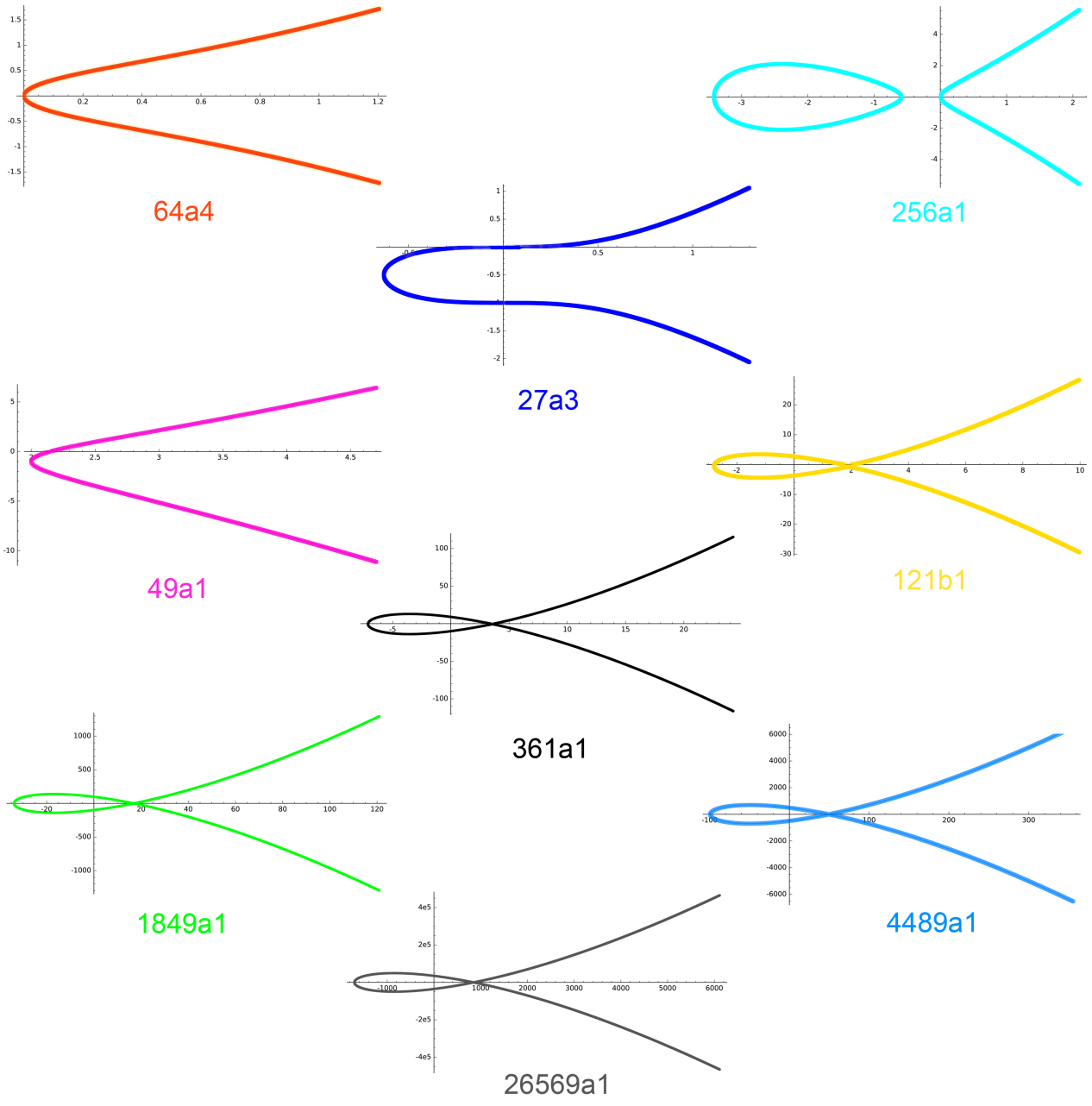

Figure 5

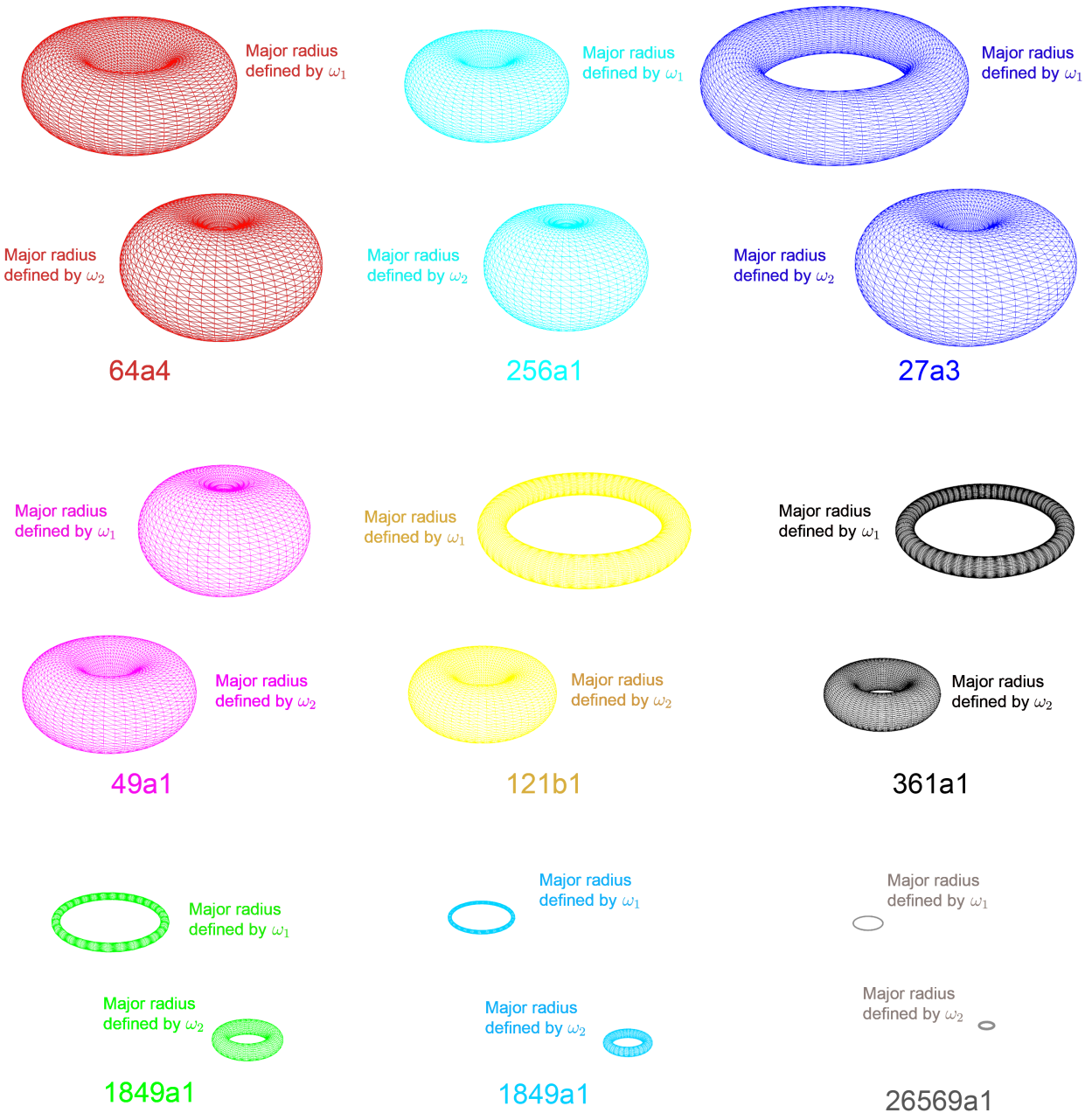
Major radius defined by ω_1
Major radius defined by ω_2
64a4
Major radius defined by ω_1
Major radius defined by ω_2
256a1
Major radius defined by ω_1
Major radius defined by ω_2
27a3
Major radius defined by ω_1
Major radius defined by ω_2
49a1
Major radius defined by ω_1
Major radius defined by ω_2
121b1
Major radius defined by ω_1
Major radius defined by ω_2
361a1
Major radius defined by ω_1
Major radius defined by ω_2
1849a1
Major radius defined by ω_1
Major radius defined by ω_2
1849a1
Major radius defined by ω_1
Major radius defined by ω_2
26569a1

Figure 6

6 Open Questions

There is currently no definitive method for calculating the rank of an elliptic curve. More specifically, it is unknown whether the rank of an elliptic curve can be arbitrarily large (i.e. whether ranks are bounded or unbounded). The Birch and Swinnerton-Dyer conjecture, a member of the million dollar Millenn'ium Problems, gives a method for finding the rank of the group $E(K)$, but has only been proven in some special cases including particular CM curves. See, for example, [4]. Currently, the largest known rank is *at least* 24, discovered by Martin and McMillen in 2000.

7 Acknowledgements

Thank you to Professor Alina Cojocaru and Cara Mullen for your time and dedication in trying to teach an art student and a history major math! Thank you to Professor Benjamin Antieau, Professor David Dumas, and everyone else helping to make the the Mathematical Computing Lab at UIC the great place that it is. This project would not have been possible without the lab's resources.

Appendix: Sage Code

```
def latticeHeight(lattice):
    w1, w2 = lattice.basis();
    u = vector([w1.real(),w1.imag()]);
    v = vector([w2.real(), w2.imag()]);
    cosAngle = u.dot_product(v) / (w1.abs()*w2.abs());
    sinAngle = sqrt(1 - cosAngle^2)
    h1 = w2.abs()*sinAngle
    h2 = w1.abs()*sinAngle
    return (h1, h2)

def latticeRhombus(lattice):
    w1, w2 = lattice.basis()
    rhombus = polygon([(0,0),
                       (w2.real(),w2.imag()),
                       (w1.real() + w2.real(), w2.imag()),
                       (w1.real(),w1.imag())])
    return rhombus

def latticeTori(lattice):
    h1, h2 = latticeHeight(lattice)
    w1, w2 = lattice.basis()
    R1 = w1.abs() / (2*pi)
    r1 = h1 / (2*pi)
    R2 = w2.abs() / (2*pi)
```

```
    r2 = h2 / (2*pi)
    return {"Torus1": (surfaces.Torus(r1, R1), R1, r1),
            "Torus2": (surfaces.Torus(r2, R2), R2, r2)}

def curveData(EC):
    K = RationalField()
    R = K.embeddings(RealField())[0]
    lattice = EC.period_lattice(R)
    rhombus = latticeRhombus(lattice)
    tori = latticeTori(lattice)
    return (EC, lattice, rhombus, tori)

E1 = curveData(EllipticCurve([0,0,0,1,0]))
E2 = curveData(EllipticCurve([0,4,0,2,0]))
E3 = curveData(EllipticCurve([0,0,1,0,0]))
E4 = curveData(EllipticCurve([1,-1,0,-2,-1]))
E5 = curveData(EllipticCurve([0,-1,1,-7,10]))
E6 = curveData(EllipticCurve([0,0,1,-38,90]))
E7 = curveData(EllipticCurve([0,0,1,-860,9707]))
E8 = curveData(EllipticCurve([0,0,1,-7370,243528]))
E9 = curveData(EllipticCurve([0,0,1,-2174420,1234136692]))

print "#1: " + str(E1[0])
plot(E1[0])
print "Weierstrauss Form:"
E1[0].short_weierstrass_model()
print "\nDiscriminant: " + str(E1[0].discriminant()) + "\n"
print "Conductor: " + str(E1[0].conductor()) + "\n"
print "j-invariant: " + str(E1[0].j_invariant()) + "\n"
print "Rank: " + str(E1[0].rank()) + "\n"
print "Torsion Subgroup: " + str(E1[0].torsion_subgroup()) + "\n"
print "Real/Complex Embedding - Period Lattice basis: " + str(E1[1].basis()) + "\n"
print "Rhombus:"
E1[2]
print ("\nRhombus height 1 (orthogonal to omega 1): "
       + str(latticeHeight(E1[1])[0]) + "\n")
print "Torus 1A (Major Radius from omega 1):"
E1[3]["Torus1"][0].plot()
print "\nMajor Radius: " + str(E1[3]["Torus1"][1]) + "\n"
print "Minor Radius: " + str(E1[3]["Torus1"][2]) + "\n"
print ("Rhombus height 2 (orthogonal to omega 2): "
       + str(latticeHeight(E1[1])[1]) + "\n")
print "Torus 1B (Major Radius from omega 2):"
E1[3]["Torus2"][0].plot()
print "\nMajor Radius: " + str(E1[3]["Torus2"][1]) + "\n"
```

```
print "Minor Radius: " + str(E1[3]["Torus2"][2]) + "\n\n\n"
print "End #1 ************************************************************\n\n\n"

E2 = curveData(EllipticCurve([0,4,0,2,0]))
print "#2: " + str(E2[0])
plot(E2[0])
print "Weierstrauss Form:"
E2[0].short_weierstrass_model()
print "\nDiscriminant: " + str(E2[0].discriminant()) + "\n"
print "Conductor: " + str(E2[0].conductor()) + "\n"
print "j-invariant: " + str(E2[0].j_invariant()) + "\n"
print "Rank: " + str(E2[0].rank()) + "\n"
print "Torsion Subgroup: " + str(E2[0].torsion_subgroup()) + "\n"
print "Real/Complex Embedding - Period Lattice basis: " + str(E2[1].basis()) +"\n"
print "Rhombus:"
E2[2]
print ("\nRhombus height 1 (orthogonal to omega 1): "
       + str(latticeHeight(E2[1])[0]) + "\n")
print "Torus 2A (Major Radius from omega 1):"
E2[3]["Torus1"][0].plot()
print "\nMajor Radius: " + str(E2[3]["Torus1"][1]) + "\n"
print "Minor Radius: " + str(E2[3]["Torus1"][2]) + "\n"
print ("Rhombus height 2 (orthogonal to omega 2): "
       + str(latticeHeight(E2[1])[1]) + "\n")
print "Torus 2B (Major Radius from omega 2):"
E2[3]["Torus2"][0].plot()
print "\nMajor Radius: " + str(E2[3]["Torus2"][1]) + "\n"
print "Minor Radius: " + str(E2[3]["Torus2"][2]) + "\n\n\n"
print "End #2 ************************************************************\n\n\n"

E3 = curveData(EllipticCurve([0,0,1,0,0]))
print "#3: " + str(E3[0])
plot(E3[0])
print "Weierstrauss Form:"
E3[0].short_weierstrass_model()
print "\nDiscriminant: " + str(E3[0].discriminant()) + "\n"
print "Conductor: " + str(E3[0].conductor()) + "\n"
print "j-invariant: " + str(E3[0].j_invariant()) + "\n"
print "Rank: " + str(E3[0].rank()) + "\n"
print "Torsion Subgroup: " + str(E3[0].torsion_subgroup()) + "\n"
print "Real/Complex Embedding - Period Lattice basis: " + str(E3[1].basis()) +"\n"
print "Rhombus:"
E3[2]
print ("\nRhombus height 1 (orthogonal to omega 1): "
       + str(latticeHeight(E3[1])[0]) + "\n")
```

```
print "Torus 3A (Major Radius from omega 1):"
E3[3]["Torus1"][0].plot()
print "\nMajor Radius: " + str(E3[3]["Torus1"][1]) + "\n"
print "Minor Radius: " + str(E3[3]["Torus1"][2]) + "\n"
print ("Rhombus height 2 (orthogonal to omega 2): "
       + str(latticeHeight(E3[1])[1]) + "\n")
print "Torus 3B (Major Radius from omega 2):"
E3[3]["Torus2"][0].plot()
print "\nMajor Radius: " + str(E3[3]["Torus2"][1]) + "\n"
print "Minor Radius: " + str(E3[3]["Torus2"][2]) + "\n\n\n"
print "End #3 ************************************************************\n\n\n"

E4 = curveData(EllipticCurve([1,-1,0,-2,-1]))
print "#4: " + str(E4[0])
plot(E4[0])
print "Weierstrauss Form:"
E4[0].short_weierstrass_model()
print "\nDiscriminant: " + str(E4[0].discriminant()) + "\n"
print "Conductor: " + str(E4[0].conductor()) + "\n"
print "j-invariant: " + str(E4[0].j_invariant()) + "\n"
print "Rank: " + str(E4[0].rank()) + "\n"
print "Torsion Subgroup: " + str(E4[0].torsion_subgroup()) + "\n"
print "Real/Complex Embedding - Period Lattice basis: " + str(E4[1].basis()) +"\n"
print "Rhombus:"
E4[2]
print ("\nRhombus height 1 (orthogonal to omega 1): "
       + str(latticeHeight(E4[1])[0]) + "\n")
print "Torus 4A (Major Radius from omega 1):"
E4[3]["Torus1"][0].plot()
print "\nMajor Radius: " + str(E4[3]["Torus1"][1]) + "\n"
print "Minor Radius: " + str(E4[3]["Torus1"][2]) + "\n"
print ("Rhombus height 2 (orthogonal to omega 2): "
       + str(latticeHeight(E4[1])[1]) + "\n")
print "Torus 4B (Major Radius from omega 2):"
E4[3]["Torus2"][0].plot()
print "\nMajor Radius: " + str(E4[3]["Torus2"][1]) + "\n"
print "Minor Radius: " + str(E4[3]["Torus2"][2]) + "\n\n\n"
print "End #4 ************************************************************\n\n\n"

E5 = curveData(EllipticCurve([0,-1,1,-7,10]))
print "#5: " + str(E5[0])
plot(E5[0])
print "Weierstrauss Form:"
E5[0].short_weierstrass_model()
print "\nDiscriminant: " + str(E5[0].discriminant()) + "\n"
```

```
print "Conductor: " + str(E5[0].conductor()) + "\n"
print "j-invariant: " + str(E5[0].j_invariant()) + "\n"
print "Rank: " + str(E5[0].rank()) + "\n"
print "Torsion Subgroup: " + str(E5[0].torsion_subgroup()) + "\n"
print "Real/Complex Embedding - Period Lattice basis: " + str(E5[1].basis()) +"\n"
print "Rhombus:"
E5[2]
print ("\nRhombus height 1 (orthogonal to omega 1): "
       + str(latticeHeight(E5[1])[0]) + "\n")
print "Torus 5A (Major Radius from omega 1):"
E5[3]["Torus1"][0].plot()
print "\nMajor Radius: " + str(E5[3]["Torus1"][1]) + "\n"
print "Minor Radius: " + str(E5[3]["Torus1"][2]) + "\n"
print ("Rhombus height 2 (orthogonal to omega 2): "
       + str(latticeHeight(E5[1])[1]) + "\n")
print "Torus 5B (Major Radius from omega 2):"
E5[3]["Torus2"][0].plot()
print "\nMajor Radius: " + str(E5[3]["Torus2"][1]) + "\n"
print "Minor Radius: " + str(E5[3]["Torus2"][2]) + "\n\n\n"
print "End #5 ***************************************************************\n\n\n"

E6 = curveData(EllipticCurve([0,0,1,-38,90]))
print "#6: " + str(E6[0])
plot(E6[0])
print "Weierstrauss Form:"
E6[0].short_weierstrass_model()
print "\nDiscriminant: " + str(E6[0].discriminant()) + "\n"
print "Conductor: " + str(E6[0].conductor()) + "\n"
print "j-invariant: " + str(E6[0].j_invariant()) + "\n"
print "Rank: " + str(E6[0].rank()) + "\n"
print "Torsion Subgroup: " + str(E6[0].torsion_subgroup()) + "\n"
print "Real/Complex Embedding - Period Lattice basis: " + str(E6[1].basis()) +"\n"
print "Rhombus:"
E6[2]
print ("\nRhombus height 1 (orthogonal to omega 1): "
       + str(latticeHeight(E6[1])[0]) + "\n")
print "Torus 6A (Major Radius from omega 1):"
E6[3]["Torus1"][0].plot()
print "\nMajor Radius: " + str(E6[3]["Torus1"][1]) + "\n"
print "Minor Radius: " + str(E6[3]["Torus1"][2]) + "\n"
print ("Rhombus height 2 (orthogonal to omega 2): "
       + str(latticeHeight(E6[1])[1]) + "\n")
print "Torus 6B (Major Radius from omega 2):"
E6[3]["Torus2"][0].plot()
print "\nMajor Radius: " + str(E6[3]["Torus2"][1]) + "\n"
```

```
print "Minor Radius: " + str(E6[3]["Torus2"][2]) + "\n\n\n"
print "End #6 ************************************************************\n\n\n"

E7 = curveData(EllipticCurve([0,0,1,-860,9707]))
print "#7: " + str(E7[0])
plot(E7[0])
print "Weierstrauss Form:"
E7[0].short_weierstrass_model()
print "\nDiscriminant: " + str(E7[0].discriminant()) + "\n"
print "Conductor: " + str(E7[0].conductor()) + "\n"
print "j-invariant: " + str(E7[0].j_invariant()) + "\n"
print "Rank: " + str(E7[0].rank()) + "\n"
print "Torsion Subgroup: " + str(E7[0].torsion_subgroup()) + "\n"
print "Real/Complex Embedding - Period Lattice basis: " + str(E7[1].basis()) +"\n"
print "Rhombus:"
E7[2]
print ("\nRhombus height 1 (orthogonal to omega 1): "
       + str(latticeHeight(E7[1])[0]) + "\n")
print "Torus 7A (Major Radius from omega 1):"
E7[3]["Torus1"][0].plot()
print "\nMajor Radius: " + str(E7[3]["Torus1"][1]) + "\n"
print "Minor Radius: " + str(E7[3]["Torus1"][2]) + "\n"
print ("Rhombus height 2 (orthogonal to omega 2): "
       + str(latticeHeight(E7[1])[1]) + "\n")
print "Torus 7B (Major Radius from omega 2):"
E7[3]["Torus2"][0].plot()
print "\nMajor Radius: " + str(E7[3]["Torus2"][1]) + "\n"
print "Minor Radius: " + str(E7[3]["Torus2"][2]) + "\n\n\n"
print "End #7 ************************************************************\n\n\n"

E8 = curveData(EllipticCurve([0,0,1,-7370,243528]))
print "#8: " + str(E8[0])
plot(E8[0])
print "Weierstrauss Form:"
E8[0].short_weierstrass_model()
print "\nDiscriminant: " + str(E8[0].discriminant()) + "\n"
print "Conductor: " + str(E8[0].conductor()) + "\n"
print "j-invariant: " + str(E8[0].j_invariant()) + "\n"
print "Rank: " + str(E8[0].rank()) + "\n"
print "Torsion Subgroup: " + str(E8[0].torsion_subgroup()) + "\n"
print "Real/Complex Embedding - Period Lattice basis: " + str(E8[1].basis()) +"\n"
print "Rhombus:"
E8[2]
print ("\nRhombus height 1 (orthogonal to omega 1): "
       + str(latticeHeight(E8[1])[0]) + "\n")
```

```
print "Torus 8A (Major Radius from omega 1):"
E8[3]["Torus1"][0].plot()
print "\nMajor Radius: " + str(E8[3]["Torus1"][1]) + "\n"
print "Minor Radius: " + str(E8[3]["Torus1"][2]) + "\n"
print ("Rhombus height 2 (orthogonal to omega 2): "
       + str(latticeHeight(E8[1])[1]) + "\n")
print "Torus 8B (Major Radius from omega 2):"
E8[3]["Torus2"][0].plot()
print "\nMajor Radius: " + str(E8[3]["Torus2"][1]) + "\n"
print "Minor Radius: " + str(E8[3]["Torus2"][2]) + "\n\n\n"
print "End #8 ************************************************************\n\n\n"

E9 = curveData(EllipticCurve([0,0,1,-2174420,1234136692]))
print "#9: " + str(E9[0])
plot(E9[0])
print "Weierstrauss Form:"
E9[0].short_weierstrass_model()
print "\nDiscriminant: " + str(E9[0].discriminant()) + "\n"
print "Conductor: " + str(E9[0].conductor()) + "\n"
print "j-invariant: " + str(E9[0].j_invariant()) + "\n"
print "Rank: " + str(E9[0].rank()) + "\n"
print "Torsion Subgroup: " + str(E9[0].torsion_subgroup()) + "\n"
print "Real/Complex Embedding - Period Lattice basis: " + str(E9[1].basis()) +"\n"
print "Rhombus:"
E9[2]
print ("\nRhombus height 1 (orthogonal to omega 1): "
       + str(latticeHeight(E9[1])[0]) + "\n")
print "Torus 9A (Major Radius from omega 1):"
E9[3]["Torus1"][0].plot()
print "\nMajor Radius: " + str(E9[3]["Torus1"][1]) + "\n"
print "Minor Radius: " + str(E9[3]["Torus1"][2]) + "\n"
print ("Rhombus height 2 (orthogonal to omega 2): "
       + str(latticeHeight(E9[1])[1]) + "\n")
print "Torus 9B (Major Radius from omega 2):"
E9[3]["Torus2"][0].plot()
print "\nMajor Radius: " + str(E9[3]["Torus2"][1]) + "\n"
print "Minor Radius: " + str(E9[3]["Torus2"][2]) + "\n\n\n"
print "End #9 ************************************************************\n\n\n"
```

References

[1] A.C. Cojocaru, *Questions about the reductions modulo primes of an elliptic curve*, Proceedings of the 7^{th} conference of the Canadian Number Theory Association (Montreal,

2002), ed. E. Goren and H. Kisilevsky, CRM Proceedings and Lecture Notes, 36 (2004), 61-79.

[2] A.C. Cojocaru, *Primes, Elliptic Curves, and Cyclic Groups: A Synopsis*, Submitted (2016).

[3] J.E. Cremona, T. Thongjunthug, *The complex AGM, periods of elliptic cruves over* $\mathbb{C}$ *and complex elliptic logarithms*, Journal of Number Theory, 133 (2013), no. 8, 2813-2841.

[4] Y. Li, Y. Liu, and Y. Tian, *On The Birch and Sinnerton-Dyer Conjecture for CM Elliptic Curves over* $\mathbb{Q}$, arXiv:1605.01481 (2016).

[5] K. Rubin and A. Silverberg, *Ranks of Elliptic Curves*, Bulletin of the American Mathematical Society, 39 (2002), no. 4, 455-474.

[6] J. Siverman and J. Tate, Rational Points on Elliptic Curves, Undergraduate Texts in Mathematics, 2015.

[7] L.C. Washington, Eliptic Curves: Number Theory and Cryptography, Chapman & Hall/CRC, 2008.

[8] A. Wiles, *The Birch and Swinnerton-Dyer Conjecture*, Clay Mathematics Institute Web Site, `http://www.claymath.org/sites/default/files/birchswin.pdf`.