Computational techniques for verifying the Birch and Swinnerton-Dyer conjecture for curves over cubic number fields

Alex Cowan (the.alex.cowan@gmail.com)

September 4, 2012

1 Introduction and summary

This document summarizes my summer's work for prof. Henri Darmon. It is intended primarily as a guide to any other students interested in doing similar work. The bulk of my project was writing code to test the Birch and Swinnerton-Dyer conjecture for elliptic curves over cubic number fields. In this report I document this code and discuss its effectiveness. Pseudo-code will be used throughout the text in an effort to clarify some ideas. I used Sage (which runs on python) when implementing them, so if a reader thinks the pseudo-code is incomplete or too brief they can look at my code instead.

The Birch and Swinnerton-Dyer conjecture (BSD) states that if a certain complex valued L function vanishes at $s = 1$ then the elliptic curve related to it will have infinitely many points on it. My project is in two parts. First I was given a series of curves for which the L function vanished and would try to show that the curve indeed had infinitely many points on it. For the second part of the project I would try to generate curves which had infinitely many points on them, and then compute their L functions to check if they were indeed 0 at $s = 1$. Understanding the details of my work requires a rudimentary knowledge of the theory of elliptic curves. I plan to write a quick introduction to the topic based on a talk I gave to my peers during the summer.

I applied three techniques in an attempt to solve the first problem of finding points. The best of these three was able to verify BSD in about half the cases it was given, though the runtimes involved were long.

The second problem involved first finding curves which had infinitely many points on them. I have written an program which is able most of the time to succeed at this, but unfortunately I have not had time to go on to checking that their L functions vanish as appropriate.

2 Point Finding

Prof. Fearnley and prof. Kisilevsky have calculated $L(E, \chi, 1)$ for several elliptic curves E and hundreds of cubic twists given by χ. The Birch and Swinnerton-Dyer conjecture (BSD) states that if $L(E, \chi, 1) = 1$ then the rank of the twisted curve E is positive. I attempt to verify that the rank of the twisted curve is indeed positive whenever prof. Fearnley and prof. Kisilevsky calculate that $L(E, \chi, 1) = 0$. [1] This is equivalent to verifying that $\text{rank}_{\mathbb{Z}} E(K) > \text{rank}_{\mathbb{Z}} E(\mathbb{Q})$, where K is the fixed field of χ.

I employ three techniques to calculate $\text{rank}_{\mathbb{Z}} E(K)$:

1. **Brute force search**: Say that the Weierstrass form of E is given as $E : F(x, y) = 0$. For a fixed value of $x_0 \in K$, one can determine fairly efficiently whether or not there is a $y \in K$ such that $F(x_0, y) = 0$

[1] In practice $|L(E, \chi, 1)| < 10^{-14}$

by considering F as a univariate polynomial in y, for which many root finding techniques are available. The strategy of this approach is to iterate through a list of x_0 values to fix, and for each check if $F(x_0, y)$ has any roots. If indeed $F(x_0, y) = 0$ and either $x_0 \in K\backslash\mathbb{Q}$ or $y \in K\backslash\mathbb{Q}$ then, if (x_0, y) as a point on E has infinite order I suspect that $\text{rank}_{\mathbb{Z}}E(K) > \text{rank}_{\mathbb{Z}}E(\mathbb{Q})$, at least most of the time. However at this time I have no results of this nature.

This technique is occasionally successful for K of small conductor, but typically it is difficult to find points otherwise.

2. **Descent with Magma**: The math software Magma includes functions which perform various descents on elliptic curves over arbitrary number fields, returning bounds on $\text{rank}_{\mathbb{Z}}E(K)$. In particular it contains a function "MordellWeilShaInformation" which is described as applying the various descent functions in the most effective order. I run this function on twists with positive rank.

The effectiveness of this technique seems to depend heavily on the curve E on which it is being run. Some curves take a very long time to perform descent on, but the rank is often found exactly, while others run orders of magnitude more quickly, but return only loose bounds on the rank.

3. **Sieving with projections**: This technique first finds the set $\overline{X}_S$ of values $\bar{x} \in \mathcal{O}_K/I$ for which there is a $\bar{y} \in \mathcal{O}_K/(m)$ such that $\overline{F}(\bar{x}, \bar{y}) \equiv 0$, where $\mathcal{O}_K$ is the ring of integers of K and I is any ideal of $\mathcal{O}_k$. It then begins generating the preimage X_S of $\overline{X}_S$ in K, and for each x in X_S checks if $F(x, y)$, now viewed as a univariate polynomial in y, has a zero. Since X_S is a subset of K fewer values of x need to be checked than in the brute force search (which checks all of them). This process could easily be made more effective by computing $\overline{X}_S(I_1), \overline{X}_S(I_2), \ldots$ for various ideals I_1, I_2 of $\mathcal{O}_K$ and then taking x from $X_S(I_1) \cap X_S(I_2) \cap \ldots$.

However I was not able to implement this technique effectively. In my experience generating the preimage X_S from $\overline{X}_S$ was more time consuming than running a brute force search on the same range. It's conceivable that this technique may become more effective with more optimization or larger search ranges.

2.1 Results

2.1.1 Brute force search

Presented below are the results of a brute force search on the curves $E20A1$, $E37A1$, and $E40A1$. the five columns indicate the following information:

Conductor The conductor of K

Character A character which fixes K. Which character is chosen systematically, as will be explained later. χ_p is the character that sends the least positive primitive root of $(\mathbb{Z}/p\mathbb{Z})^*$ to ζ_3.

Characteristic polynomial The characteristic polynomial of K.

Result Verified if BSD has been checked for this case.

New point A point $(x, y) \in K^2\backslash\mathbb{Q}^2$ of infinite order, demonstrating that $\text{rank}_{\mathbb{Z}}E(K) > \text{rank}_{\mathbb{Z}}E(\mathbb{Q})$.

E20A2

Table 1: Results for a brute force search on $E20A2$ with depth 200 up to and including conductor 819 and depth 100 thereafter.

Conductor	Character	Characteristic polynomial	Result	New point
9	χ_9	$x^3 - 3x + 1$	Verified	$(4\alpha^2 + 4\alpha, 20\alpha^2 + 20\alpha - 14)$
63	$\chi_9\chi_7$	$x^3 - 21x - 28$	Verified	$(\frac{1}{7}\alpha, \frac{1}{7}\alpha - \frac{2}{7})$

Continued on next page

E20A2 Brute force search results – continued from previous page

Conductor	Character	Characteristic polynomial	Result	New point
73	χ_{73}	$x^3+x^2-24x-27$	No point found	–
91	$\chi_7^2\chi_{13}$	$x^3-x^2-30x-27$	No point found	–
117	$\chi_9^2\chi_{13}$	$x^3-39x+26$	Verified	$(\frac{3}{4}\alpha^2-4\alpha+\frac{9}{4}, \frac{41}{8}\alpha^2-\frac{143}{4}\alpha+\frac{177}{8})$
133	$\chi_7^2\chi_{19}$	$x^3-x^2-44x-69$	No point found	–
171	$\chi_9^2\chi_{19}$	$x^3-57x+152$	No point found	–
229	χ_{229}	$x^3+x^2-76x-212$	No point found	–
259	$\chi_7\chi_{37}$	$x^3-x^2-86x-48$	No point found	–
277	χ_{277}	$x^3+x^2-92x+236$	No point found	–
307	χ_{307}	$x^3+x^2-102x-216$	No point found	–
559	$\chi_{13}^2\chi_{43}$	$x^3-x^2-186x+207$	No point found	–
613	χ_{613}	$x^3+x^2-204x+999$	No point found	–
703	$\chi_{19}\chi_{37}$	$x^3-x^2-234x+729$	Verified	$(\frac{1}{4}\alpha^2+2\alpha+4, \frac{23}{8}\alpha^2+23\alpha-\frac{1217}{8})$
711	$\chi_9\chi_{79}$	$x^3-237x-1027$	No point found	–
727	χ_{727}	$x^3+x^2-242x+1104$	No point found	–
763	$\chi_7^2\chi_{109}$	$x^3-x^2-254x-1413$	No point found	–
819	$\chi_9\chi_7\chi_{13}$	$x^3-273x+91$	No point found	–
829	χ_{829}	$x^3+x^2-276x-307$	No point found	–
871	$\chi_{13}\chi_{67}$	$x^3-x^2-290x-1613$	No point found	–
889	$\chi_7^2\chi_{127}$	$x^3-x^2-296x+1317$	No point found	–
919	χ_{919}	$x^3+x^2-306x-1872$	No point found	–
973	$\chi_7\chi_{139}$	$x^3-x^2-324x+36$	No point found	–
1027	$\chi_{13}^2\chi_{79}$	$x^3-x^2-342x-2016$	No point found	–
1143	$\chi_9\chi_{127}$	$x^3-381x-127$	No point found	–
1333	$\chi_{31}^2\chi_{43}$	$x^3-x^2-444x-1728$	No point found	–
1339	$\chi_{13}^2\chi_{103}$	$x^3-x^2-446x+3769$	No point found	–
1359	$\chi_9\chi_{151}$	$x^3-453x+3473$	No point found	–
1399	χ_{1399}	$x^3+x^2-466x+3368$	No point found	–
1477	$\chi_7^2\chi_{211}$	$x^3-x^2-492x+3501$	No point found	–
1729	$\chi_7^2\chi_{13}\chi_{19}$	$x^3+x^2-576x-64$	No point found	–
1737	$\chi_9\chi_{193}$	$x^3-579x-4825$	No point found	–
1789	χ_{1789}	$x^3+x^2-596x-5632$	No point found	–
1933	χ_{1933}	$x^3+x^2-644x+4224$	No point found	–
1957	$\chi_{19}\chi_{103}$	$x^3-x^2-652x-6016$	No point found	–
2169	$\chi_9^2\chi_{241}$	$x^3-723x+3374$	No point found	–
2179	χ_{2179}	$x^3+x^2-726x-7344$	No point found	–
2223	$\chi_9\chi_{13}\chi_{19}$	$x^3-741x-4940$	No point found	–
2331	$\chi_9\chi_7\chi_{37}$	$x^3-777x+8029$	No point found	–
2383	χ_{2383}	$x^3+x^2-794x-2736$	No point found	–
2709	$\chi_9^2\chi_7\chi_{43}$	$x^3-903x-3311$	No point found	–
2817	$\chi_9^2\chi_{313}$	$x^3-939x-6886$	No point found	–
2977	$\chi_{13}^2\chi_{229}$	$x^3-x^2-992x+441$	No point found	–
2983	$\chi_{19}\chi_{157}$	$x^3-x^2-994x+11711$	No point found	–
3037	χ_{3037}	$x^3+x^2-1012x-5849$	No point found	–
3097	$\chi_{19}\chi_{163}$	$x^3-x^2-1032x-11241$	No point found	–
3133	$\chi_{13}\chi_{241}$	$x^3-x^2-1044x+3249$	No point found	–
3229	χ_{3229}	$x^3+x^2-1076x-5860$	No point found	–
3241	$\chi_7^2\chi_{463}$	$x^3-x^2-1080x-9603$	No point found	–
3391	χ_{3391}	$x^3+x^2-1130x+14192$	No point found	–
3627	$\chi_9^2\chi_{13}^2\chi_{31}$	$x^3-1209x-2015$	No point found	–
3631	χ_{3631}	$x^3+x^2-1210x-10624$	No point found	–
3913	$\chi_7\chi_{13}\chi_{43}$	$x^3+x^2-1304x-1884$	No point found	–

Continued on next page

E20A2 Brute force search results – continued from previous page

Conductor	Character	Characteristic polynomial	Result	New point
4093	χ_{4093}	$x^3+x^2-1364x-19707$	No point found	–
4221	$\chi_9^2\chi_7\chi_{67}$	$x^3-1407x+469$	No point found	–
4237	$\chi_{19}^2\chi_{223}$	$x^3-x^2-1412x-10671$	No point found	–
4249	$\chi_7^2\chi_{607}$	$x^3-x^2-1416x-5508$	No point found	–

E24A6

Table 2: Results for a brute force search on $E24A6$ with depth 200.

Conductor	Character	Characteristic polynomial	Result	New point
31	χ_{31}	$x^3+x^2-10x-8$	No point found	–
67	χ_{67}	$x^3+x^2-22x+5$	No point found	–
133	$\chi_7\chi_{19}$	$x^3-x^2-44x+64$	No point found	–
151	χ_{151}	$x^3+x^2-50x-123$	No point found	–
193	χ_{193}	$x^3+x^2-64x+143$	No point found	–
247	$\chi_{13}^2\chi_{19}$	$x^3-x^2-82x+64$	No point found	–
469	$\chi_7\chi_{67}$	$x^3-x^2-156x-608$	No point found	–
547	χ_{547}	$x^3+x^2-182x-81$	No point found	–
589	$\chi_{19}^2\chi_{31}$	$x^3-x^2-196x+349$	No point found	–
613	χ_{613}	$x^3+x^2-204x+999$	No point found	–
679	$\chi_7\chi_{97}$	$x^3-x^2-226x+176$	No point found	–
691	χ_{691}	$x^3+x^2-230x+128$	No point found	–
703	$\chi_{19}\chi_{37}$	$x^3-x^2-234x+729$	No point found	–
739	χ_{739}	$x^3+x^2-246x-520$	No point found	–
817	$\chi_{19}^2\chi_{43}$	$x^3-x^2-272x+1755$	No point found	–
853	χ_{853}	$x^3+x^2-284x+1011$	No point found	–
871	$\chi_{13}^2\chi_{67}$	$x^3-x^2-290x+1000$	No point found	–
1009	χ_{1009}	$x^3+x^2-336x-1719$	No point found	–
1057	$\chi_7\chi_{151}$	$x^3-x^2-352x-1840$	No point found	–
1147	$\chi_{31}\chi_{37}$	$x^3-x^2-382x+2209$	No point found	–
1159	$\chi_{19}^2\chi_{61}$	$x^3-x^2-386x-1760$	No point found	–
1261	$\chi_{13}^2\chi_{97}$	$x^3-x^2-420x+1728$	No point found	–
1333	$\chi_{31}\chi_{43}$	$x^3-x^2-444x+3604$	No point found	–
1351	$\chi_7\chi_{193}$	$x^3-x^2-450x+2752$	No point found	–
1423	χ_{1423}	$x^3+x^2-474x+896$	No point found	–
1531	χ_{1531}	$x^3+x^2-510x-567$	No point found	–
1621	χ_{1621}	$x^3+x^2-540x-4923$	No point found	–
1723	χ_{1723}	$x^3+x^2-574x-2744$	No point found	–
1729	$\chi_7\chi_{13}\chi_{19}$	$x^3+x^2-576x+1665$	No point found	–
1783	χ_{1783}	$x^3+x^2-594x+5283$	No point found	–
1789	χ_{1789}	$x^3+x^2-596x-5632$	No point found	–
1843	$\chi_{19}^2\chi_{97}$	$x^3-x^2-614x-5256$	No point found	–
1861	χ_{1861}	$x^3+x^2-620x-2757$	No point found	–
1867	χ_{1867}	$x^3+x^2-622x-6085$	No point found	–
1879	χ_{1879}	$x^3+x^2-626x-5289$	No point found	–
2017	χ_{2017}	$x^3+x^2-672x-2764$	No point found	–
2071	$\chi_{19}^2\chi_{109}$	$x^3-x^2-690x-3375$	No point found	–

Continued on next page

E24A6 Brute force search results – continued from previous page

Conductor	Character	Characteristic polynomial	Result	New point
2641	$\chi_{19}\chi_{139}$	$x^3 - x^2 - 880x + 6847$	No point found	–
2653	$\chi_7\chi_{379}$	$x^3 - x^2 - 884x + 8352$	No point found	–
2749	χ_{2749}	$x^3 + x^2 - 916x + 1120$	No point found	–
2803	χ_{2803}	$x^3 + x^2 - 934x + 9551$	No point found	–
2821	$\chi_7^2\chi_{13}^2\chi_{31}$	$x^3 + x^2 - 940x - 2612$	No point found	–
2983	$\chi_{19}^2\chi_{157}$	$x^3 - x^2 - 994x - 221$	No point found	–
3199	$\chi_7^2\chi_{457}$	$x^3 - x^2 - 1066x + 12559$	No point found	–
3241	$\chi_7\chi_{463}$	$x^3 - x^2 - 1080x + 13084$	No point found	–
3343	χ_{3343}	$x^3 + x^2 - 1114x + 8048$	No point found	–
3439	$\chi_{19}^2\chi_{181}$	$x^3 - x^2 - 1146x - 11336$	No point found	–
3529	χ_{3529}	$x^3 + x^2 - 1176x + 5751$	No point found	–
3589	$\chi_{37}\chi_{97}$	$x^3 - x^2 - 1196x + 12495$	No point found	–
3679	$\chi_{13}^2\chi_{283}$	$x^3 - x^2 - 1226x - 6813$	No point found	–
3823	χ_{3823}	$x^3 + x^2 - 1274x - 14584$	No point found	–
3937	$\chi_{31}\chi_{127}$	$x^3 - x^2 - 1312x + 14144$	No point found	–
4039	$\chi_7\chi_{577}$	$x^3 - x^2 - 1346x - 16904$	No point found	–
4663	χ_{4663}	$x^3 + x^2 - 1554x + 2936$	No point found	–
4867	$\chi_{31}\chi_{157}$	$x^3 - x^2 - 1622x - 13880$	No point found	–
4891	$\chi_{67}^2\chi_{73}$	$x^3 - x^2 - 1630x + 25723$	No point found	–
5101	χ_{5101}	$x^3 + x^2 - 1700x + 17948$	No point found	–
5173	$\chi_7^2\chi_{739}$	$x^3 - x^2 - 1724x - 26823$	No point found	–
5197	χ_{5197}	$x^3 + x^2 - 1732x - 25600$	No point found	–
5227	χ_{5227}	$x^3 + x^2 - 1742x + 6195$	No point found	–

E37A1

Table 3: Results for a brute force search on $E37A1$ with depth 200.

Conductor	Character	Characteristic polynomial	Result	New point
43	χ_{43}	$x^3 + x^2 - 14x + 8$	Verified	$(\frac{1}{2}\alpha^2 - \frac{3}{2}\alpha + 1, 2\alpha^2 - 7\alpha + 3)$
61	χ_{61}	$x^3 + x^2 - 20x - 9$	Verified	$(-\frac{2}{9}\alpha, \frac{4}{81}\alpha^2 + \frac{2}{81}\alpha - \frac{8}{9})$
103	χ_{103}	$x^3 + x^2 - 34x - 61$	No point found	–
127	χ_{127}	$x^3 + x^2 - 42x + 80$	Verified	$(\frac{1}{2}\alpha^2 + \frac{7}{2}\alpha - 4, 3\alpha^2 + 18\alpha - 41)$
171	$\chi_9\chi_{19}$	$x^3 - 57x - 19$	No point found	–
247	$\chi_{13}^2\chi_{19}$	$x^3 - x^2 - 82x + 64$	Verified	$(\frac{1}{6}\alpha^2 + \frac{3}{2}\alpha + \frac{1}{3}, \alpha^2 + 8\alpha - 9)$
817	$\chi_{19}^2\chi_{43}$	$x^3 - x^2 - 272x + 1755$	No point found	–
853	χ_{853}	$x^3 + x^2 - 284x + 1011$	No point found	–
1093	χ_{1093}	$x^3 + x^2 - 364x - 1012$	No point found	–
1099	$\chi_7^2\chi_{157}$	$x^3 - x^2 - 366x - 1791$	No point found	–
1267	$\chi_7\chi_{181}$	$x^3 - x^2 - 422x + 3144$	No point found	–
1333	$\chi_{31}^2\chi_{43}$	$x^3 - x^2 - 444x - 1728$	No point found	–
1609	χ_{1609}	$x^3 + x^2 - 536x - 1311$	No point found	–
1627	χ_{1627}	$x^3 + x^2 - 542x + 4640$	No point found	–
1953	$\chi_9^2\chi_7^2\chi_{31}$	$x^3 - 651x - 5642$	No point found	–
2017	χ_{2017}	$x^3 + x^2 - 672x - 2764$	No point found	–
2263	$\chi_{31}\chi_{73}$	$x^3 - x^2 - 754x - 7711$	No point found	–
2611	$\chi_7\chi_{373}$	$x^3 - x^2 - 870x + 8800$	No point found	–
2709	$\chi_9^2\chi_7^2\chi_{43}$	$x^3 - 903x - 602$	No point found	–
2863	$\chi_7\chi_{409}$	$x^3 - x^2 - 954x + 5832$	No point found	–

Continued on next page

E37A1 Brute force search results – continued from previous page

Conductor	Character	Characteristic polynomial	Result	New point
2869	$\chi_{19}^2\chi_{151}$	$x^3 - x^2 - 956x - 2444$	No point found	–
2983	$\chi_{19}^2\chi_{157}$	$x^3 - x^2 - 994x - 221$	No point found	–
3331	χ_{3331}	$x^3 + x^2 - 1110x - 2344$	No point found	–
3379	$\chi_{31}^2\chi_{109}$	$x^3 - x^2 - 1126x + 14392$	No point found	–
3411	$\chi_9\chi_{379}$	$x^3 - 1137x - 14023$	No point found	–
3457	χ_{3457}	$x^3 + x^2 - 1152x + 13700$	No point found	–
3787	$\chi_7\chi_{541}$	$x^3 - x^2 - 1262x - 10800$	No point found	–
3829	$\chi_7\chi_{547}$	$x^3 - x^2 - 1276x - 16876$	No point found	–
4027	χ_{4027}	$x^3 + x^2 - 1342x + 15064$	No point found	–
4123	$\chi_7\chi_{19}\chi_{31}$	$x^3 + x^2 - 1374x - 13896$	No point found	–
4273	χ_{4273}	$x^3 + x^2 - 1424x + 7913$	No point found	–
4383	$\chi_9^2\chi_{487}$	$x^3 - 1461x + 21428$	No point found	–
4417	$\chi_7^2\chi_{631}$	$x^3 - x^2 - 1472x + 8016$	No point found	–
4423	χ_{4423}	$x^3 + x^2 - 1474x + 10648$	No point found	–
4687	$\chi_{43}^2\chi_{109}$	$x^3 - x^2 - 1562x + 17880$	No point found	–
4723	χ_{4723}	$x^3 + x^2 - 1574x + 21341$	No point found	–
4891	$\chi_{67}\chi_{73}$	$x^3 - x^2 - 1630x - 18296$	No point found	–
5139	$\chi_9\chi_{571}$	$x^3 - 1713x - 9136$	No point found	–
5323	χ_{5323}	$x^3 + x^2 - 1774x - 22672$	No point found	–
5383	$\chi_7^2\chi_{769}$	$x^3 - x^2 - 1794x - 17744$	No point found	–
5473	$\chi_{13}^2\chi_{421}$	$x^3 - x^2 - 1824x + 29392$	No point found	–
5517	$\chi_9^2\chi_{613}$	$x^3 - 1839x - 6130$	No point found	–
5719	$\chi_7\chi_{19}\chi_{43}$	$x^3 + x^2 - 1906x - 31984$	No point found	–
5761	$\chi_7\chi_{823}$	$x^3 - x^2 - 1920x - 30512$	No point found	–
5803	$\chi_7\chi_{829}$	$x^3 - x^2 - 1934x - 32024$	No point found	–
5917	$\chi_{61}^2\chi_{97}$	$x^3 - x^2 - 1972x - 10300$	No point found	–
6097	$\chi_7^2\chi_{13}\chi_{67}$	$x^3 + x^2 - 2032x - 21904$	No point found	–
6139	$\chi_7\chi_{877}$	$x^3 - x^2 - 2046x + 4320$	No point found	–
6381	$\chi_9^2\chi_{709}$	$x^3 - 2127x - 15598$	No point found	–
6643	$\chi_7\chi_{13}^2\chi_{73}$	$x^3 + x^2 - 2214x - 40104$	No point found	–
6751	$\chi_{43}\chi_{157}$	$x^3 - x^2 - 2250x - 40256$	No point found	–
7039	χ_{7039}	$x^3 + x^2 - 2346x + 11471$	No point found	–
7699	χ_{7699}	$x^3 + x^2 - 2566x - 32792$	No point found	–
8197	$\chi_7\chi_{1171}$	$x^3 - x^2 - 2732x - 50700$	No point found	–
8779	χ_{8779}	$x^3 + x^2 - 2926x + 32840$	No point found	–
8911	$\chi_7\chi_{19}\chi_{67}$	$x^3 + x^2 - 2970x - 62707$	No point found	–
9351	$\chi_9^2\chi_{1039}$	$x^3 - 3117x - 54028$	No point found	–
9459	$\chi_9^2\chi_{1051}$	$x^3 - 3153x + 24173$	No point found	–
9589	$\chi_{43}^2\chi_{223}$	$x^3 - x^2 - 3196x + 21664$	No point found	–
9891	$\chi_9\chi_7^2\chi_{157}$	$x^3 - 3297x - 18683$	No point found	–
9973	χ_{9973}	$x^3 + x^2 - 3324x - 26964$	No point found	–
10423	$\chi_7^2\chi_{1489}$	$x^3 - x^2 - 3474x + 10809$	No point found	–
10801	$\chi_7\chi_{1543}$	$x^3 - x^2 - 3600x + 400$	No point found	–
10963	$\chi_{19}^2\chi_{577}$	$x^3 - x^2 - 3654x + 66184$	No point found	–
11263	$\chi_7^2\chi_{1609}$	$x^3 - x^2 - 3754x - 87184$	No point found	–
11347	$\chi_7\chi_{1621}$	$x^3 - x^2 - 3782x - 32360$	No point found	–
12537	$\chi_9\chi_7^2\chi_{199}$	$x^3 - 4179x - 69650$	No point found	–
12799	χ_{12799}	$x^3 + x^2 - 4266x + 76320$	No point found	–
13333	$\chi_{67}^2\chi_{199}$	$x^3 - x^2 - 4444x - 112096$	No point found	–
13399	χ_{13399}	$x^3 + x^2 - 4466x + 91808$	No point found	–
13477	χ_{13477}	$x^3 + x^2 - 4492x + 90845$	No point found	–

Continued on next page

E37A1 Brute force search results – continued from previous page

Conductor	Character	Characteristic polynomial	Result	New point
13657	$\chi_7\chi_{1951}$	$x^3-x^2-4552x-114820$	No point found	–
14437	χ_{14437}	$x^3+x^2-4812x+37964$	No point found	–
14677	$\chi_{13}^2\chi_{1129}$	$x^3-x^2-4892x+85344$	No point found	–
15007	$\chi_{43}\chi_{349}$	$x^3-x^2-5002x+98935$	No point found	–
15067	$\chi_{13}\chi_{19}^2\chi_{61}$	$x^3+x^2-5022x+135045$	No point found	–
15673	$\chi_7^2\chi_{2239}$	$x^3-x^2-5224x+14512$	No point found	–
15723	$\chi_9^2\chi_{1747}$	$x^3-5241x+139760$	No point found	–
16263	$\chi_9\chi_{13}\chi_{139}$	$x^3-5421x+137332$	No point found	–
16267	χ_{16267}	$x^3+x^2-5422x-112664$	No point found	–
17073	$\chi_9\chi_7^2\chi_{271}$	$x^3-5691x-163142$	No point found	–
17113	$\chi_{109}^2\chi_{157}$	$x^3-x^2-5704x-143876$	No point found	–
17233	$\chi_{19}^2\chi_{907}$	$x^3-x^2-5744x+54252$	No point found	–
17937	$\chi_9\chi_{1993}$	$x^3-5979x-139510$	No point found	–
18103	$\chi_{43}\chi_{421}$	$x^3-x^2-6034x+179689$	No point found	–
18307	χ_{18307}	$x^3+x^2-6102x-110520$	No point found	–
18529	$\chi_7^2\chi_{2647}$	$x^3-x^2-6176x-164016$	No point found	–
18781	$\chi_7^2\chi_{2683}$	$x^3-x^2-6260x-97383$	No point found	–
19201	$\chi_7\chi_{13}\chi_{211}$	$x^3+x^2-6400x+167831$	No point found	–
19579	$\chi_7\chi_{2797}$	$x^3-x^2-6526x-73240$	No point found	–
19843	χ_{19843}	$x^3+x^2-6614x-207984$	No point found	–
19933	$\chi_{31}^2\chi_{643}$	$x^3-x^2-6644x+62752$	No point found	–
20167	$\chi_7^2\chi_{43}^2\chi_{67}$	$x^3+x^2-6722x-110545$	No point found	–
20529	$\chi_9\chi_{2281}$	$x^3-6843x-180199$	No point found	–
21231	$\chi_9\chi_7^2\chi_{337}$	$x^3-7077x-103796$	No point found	–
21883	$\chi_{79}\chi_{277}$	$x^3-x^2-7294x-220451$	No point found	–
22441	χ_{22441}	$x^3+x^2-7480x-140464$	No point found	–
22689	$\chi_9^2\chi_{2521}$	$x^3-7563x+65546$	No point found	–
22867	$\chi_{13}^2\chi_{1759}$	$x^3-x^2-7622x+239680$	No point found	–
23121	$\chi_9^2\chi_7\chi_{367}$	$x^3-7707x-259469$	No point found	–
23293	χ_{23293}	$x^3+x^2-7764x-125092$	No point found	–
23389	$\chi_{19}^2\chi_{1231}$	$x^3-x^2-7796x-207036$	No point found	–
23959	$\chi_{13}\chi_{19}^2\chi_{97}$	$x^3+x^2-7986x+102935$	No point found	–
24007	χ_{24007}	$x^3+x^2-8002x-198280$	No point found	–
24093	$\chi_9^2\chi_{2677}$	$x^3-8031x-187390$	No point found	–
24349	$\chi_{13}^2\chi_{1873}$	$x^3-x^2-8116x-9920$	No point found	–
24421	χ_{24421}	$x^3+x^2-8140x-22612$	No point found	–
24589	$\chi_{67}^2\chi_{367}$	$x^3-x^2-8196x+33696$	No point found	–
24943	χ_{24943}	$x^3+x^2-8314x-264211$	No point found	–
25389	$\chi_9\chi_7^2\chi_{13}\chi_{31}$	$x^3-8463x+208754$	No point found	–
25909	$\chi_{13}^2\chi_{1993}$	$x^3-x^2-8636x-186161$	No point found	–
27001	$\chi_{13}^2\chi_{31}\chi_{67}$	$x^3+x^2-9000x-217008$	No point found	–
27333	$\chi_9^2\chi_{3037}$	$x^3-9111x+334070$	No point found	–
27823	χ_{27823}	$x^3+x^2-9274x-276169$	No point found	–
28537	χ_{28537}	$x^3+x^2-9512x+328704$	No point found	–
28807	χ_{28807}	$x^3+x^2-9602x+274200$	No point found	–
29629	χ_{29629}	$x^3+x^2-9876x+156924$	No point found	–
29701	$\chi_7^2\chi_{4243}$	$x^3-x^2-9900x-325611$	No point found	–
29871	$\chi_9^2\chi_{3319}$	$x^3-9957x-252244$	No point found	–
30079	$\chi_7^2\chi_{4297}$	$x^3-x^2-10026x+61272$	No point found	–
31057	$\chi_{13}^2\chi_{2389}$	$x^3-x^2-10352x-399140$	No point found	–

E40A3

Table 4: Results for a brute force search on $E40A3$ with depth 200.

Conductor	Character	Characteristic polynomial	Result	New point
7	χ_7	x^3+x^2-2x-1	Verified	$(-\alpha,\alpha)$
63	$\chi_9\chi_7$	$x^3-21x-28$	Verified	$(\frac{1}{2}\alpha^2+\frac{1}{2}\alpha-1, \frac{1}{2}\alpha^2+\frac{13}{2}\alpha+10)$
91	$\chi_7\chi_{13}$	$x^3-x^2-30x+64$	No point found	–
223	χ_{223}	$x^3+x^2-74x-256$	No point found	–
259	$\chi_7\chi_{37}$	$x^3-x^2-86x-48$	No point found	–
277	χ_{277}	$x^3+x^2-92x+236$	No point found	–
301	$\chi_7^2\chi_{43}$	$x^3-x^2-100x+379$	No point found	–
427	$\chi_7\chi_{61}$	$x^3-x^2-142x+680$	No point found	–
499	χ_{499}	$x^3+x^2-166x+536$	No point found	–
547	χ_{547}	$x^3+x^2-182x-81$	No point found	–
619	χ_{619}	$x^3+x^2-206x+321$	No point found	–
679	$\chi_7^2\chi_{97}$	$x^3-x^2-226x-503$	No point found	–
757	χ_{757}	$x^3+x^2-252x+729$	No point found	–
853	χ_{853}	$x^3+x^2-284x+1011$	No point found	–
883	χ_{883}	$x^3+x^2-294x+1439$	No point found	–
919	χ_{919}	$x^3+x^2-306x-1872$	No point found	–
973	$\chi_7\chi_{139}$	$x^3-x^2-324x+36$	No point found	–
1009	χ_{1009}	$x^3+x^2-336x-1719$	No point found	–
1129	χ_{1129}	$x^3+x^2-376x-2927$	No point found	–
1197	$\chi_9\chi_7^2\chi_{19}$	$x^3-399x+2926$	No point found	–
1267	$\chi_7\chi_{181}$	$x^3-x^2-422x+3144$	No point found	–
1359	$\chi_9\chi_{151}$	$x^3-453x+3473$	No point found	–
1447	χ_{1447}	$x^3+x^2-482x+1715$	No point found	–
1729	$\chi_7^2\chi_{13}\chi_{19}$	$x^3+x^2-576x-64$	No point found	–
1777	χ_{1777}	$x^3+x^2-592x+724$	No point found	–
1789	χ_{1789}	$x^3+x^2-596x-5632$	No point found	–
1897	$\chi_7^2\chi_{271}$	$x^3-x^2-632x+4075$	No point found	–
1899	$\chi_9^2\chi_{211}$	$x^3-633x-3376$	No point found	–
1999	χ_{1999}	$x^3+x^2-666x-4072$	No point found	–
2007	$\chi_9^2\chi_{223}$	$x^3-669x+1115$	No point found	–
2077	$\chi_{31}\chi_{67}$	$x^3-x^2-692x+7231$	No point found	–
2131	χ_{2131}	$x^3+x^2-710x-7419$	No point found	–
2149	$\chi_7^2\chi_{307}$	$x^3-x^2-716x+6049$	No point found	–
2169	$\chi_9^2\chi_{241}$	$x^3-723x+3374$	No point found	–
2191	$\chi_7\chi_{313}$	$x^3-x^2-730x+568$	No point found	–
2353	$\chi_{13}\chi_{181}$	$x^3-x^2-784x+7669$	No point found	–
2383	χ_{2383}	$x^3+x^2-794x-2736$	No point found	–
2479	$\chi_{37}^2\chi_{67}$	$x^3-x^2-826x-5968$	No point found	–
2547	$\chi_9^2\chi_{283}$	$x^3-849x-7075$	No point found	–
2709	$\chi_9\chi_7\chi_{43}$	$x^3-903x+10234$	No point found	–
2731	χ_{2731}	$x^3+x^2-910x+10216$	No point found	–
2947	$\chi_7\chi_{421}$	$x^3-x^2-982x-11024$	No point found	–
2971	χ_{2971}	$x^3+x^2-990x+5832$	No point found	–

2.1.2 Descent with Magma

Below are the results of using Magma to perform a series of descents on the curves $E20A2$, $E24A6$, $E37A$, and $E40A3$ over various number fields. The first four columns are the same as in the previous section.

Rank bounds The bounds on $\operatorname{rank}_{\mathbb{Z}} E(K)$ returned by Magma, given as [lower, upper]

A result of "lower bound too loose" indicates that Magma was not able to find enough generators to verify BSD. In all these cases it so happens that this is exactly when the lower bound on the rank is 0, but for example if a lower bound of 1 was returned for $E37A1/K$ this description would also be appropriate since $E37A1/\mathbb{Q}$ has rank 1 already.

The generators which were found are given in the appendix.

E20A2

Table 5: Results using Magma to perform descents on $E20A2$.

Conductor	Character	Characteristic polynomial	Result	Rank bounds
9	χ_9	x^3-3x+1	Verified	$[2,2]$
63	$\chi_9\chi_7$	$x^3-21x-28$	Verified	$[2,2]$
73	χ_{73}	$x^3+x^2-24x-27$	Verified	$[2,2]$
91	$\chi_7^2\chi_{13}$	$x^3-x^2-30x-27$	Verified	$[2,2]$
117	$\chi_9^2\chi_{13}$	$x^3-39x+26$	Verified	$[2,2]$
133	$\chi_7^2\chi_{19}$	$x^3-x^2-44x-69$	Verified	$[2,2]$
171	$\chi_9^2\chi_{19}$	$x^3-57x+152$	Verified	$[2,2]$
229	χ_{229}	$x^3+x^2-76x-212$	Verified	$[2,2]$
259	$\chi_7\chi_{37}$	$x^3-x^2-86x-48$	Verified	$[2,2]$
277	χ_{277}	$x^3+x^2-92x+236$	Verified	$[2,2]$
307	χ_{307}	$x^3+x^2-102x-216$	Verified	$[2,2]$
559	$\chi_{13}^2\chi_{43}$	$x^3-x^2-186x+207$	Lower bound too loose	$[0,2]$
613	χ_{613}	$x^3+x^2-204x+999$	Lower bound too loose	$[0,2]$
703	$\chi_{19}\chi_{37}$	$x^3-x^2-234x+729$	Verified	$[4,4]$
711	$\chi_9\chi_{79}$	$x^3-237x-1027$	Lower bound too loose	$[0,2]$
727	χ_{727}	$x^3+x^2-242x+1104$	Verified	$[4,4]$
763	$\chi_7^2\chi_{109}$	$x^3-x^2-254x-1413$	Verified	$[1,2]$
819	$\chi_9\chi_7\chi_{13}$	$x^3-273x+91$	Lower bound too loose	$[0,2]$
829	χ_{829}	$x^3+x^2-276x-307$	Verified	$[2,2]$
871	$\chi_{13}\chi_{67}$	$x^3-x^2-290x-1613$	Lower bound too loose	$[0,2]$
889	$\chi_7^2\chi_{127}$	$x^3-x^2-296x+1317$	Lower bound too loose	$[0,2]$
919	χ_{919}	$x^3+x^2-306x-1872$	Lower bound too loose	$[0,2]$
973	$\chi_7\chi_{139}$	$x^3-x^2-324x+36$	Lower bound too loose	$[0,2]$
1027	$\chi_{13}^2\chi_{79}$	$x^3-x^2-342x-2016$	Verified	$[2,2]$
1143	$\chi_9\chi_{127}$	$x^3-381x-127$	Lower bound too loose	$[0,2]$
1333	$\chi_{31}^2\chi_{43}$	$x^3-x^2-444x-1728$	Verified	$[2,2]$
1339	$\chi_{13}^2\chi_{103}$	$x^3-x^2-446x+3769$	Lower bound too loose	$[0,2]$
1359	$\chi_9\chi_{151}$	$x^3-453x+3473$	Verified	$[1,2]$
1399	χ_{1399}	$x^3+x^2-466x+3368$	Lower bound too loose	$[0,2]$
1477	$\chi_7^2\chi_{211}$	$x^3-x^2-492x+3501$	Verified	$[2,2]$
1729	$\chi_7^2\chi_{13}\chi_{19}$	$x^3+x^2-576x-64$	Lower bound too loose	$[0,2]$
1737	$\chi_9\chi_{193}$	$x^3-579x-4825$	Lower bound too loose	$[0,4]$
1789	χ_{1789}	$x^3+x^2-596x-5632$	Lower bound too loose	$[0,2]$
1933	χ_{1933}	$x^3+x^2-644x+4224$	Lower bound too loose	$[0,2]$

Continued on next page

E20A2 Descent with Magma results – continued from previous page

Conductor	Character	Characteristic polynomial	Result	Rank bounds
1957	$\chi_{19}\chi_{103}$	$x^3 - x^2 - 652x - 6016$	Lower bound too loose	$[0,2]$
2169	$\chi_9^2\chi_{241}$	$x^3 - 723x + 3374$	Lower bound too loose	$[0,2]$
2179	χ_{2179}	$x^3 + x^2 - 726x - 7344$	Lower bound too loose	$[0,2]$
2223	$\chi_9\chi_{13}\chi_{19}$	$x^3 - 741x - 4940$	Lower bound too loose	$[0,2]$
2331	$\chi_9\chi_7\chi_{37}$	$x^3 - 777x + 8029$	Verified	$[1,2]$
2383	χ_{2383}	$x^3 + x^2 - 794x - 2736$	Lower bound too loose	$[0,2]$
2709	$\chi_9^2\chi_7\chi_{43}$	$x^3 - 903x - 3311$	Verified	$[1,2]$
2817	$\chi_9^2\chi_{313}$	$x^3 - 939x - 6886$	Verified	$[1,2]$

E24A6

Table 6: Results using Magma to perform descents on $E24A6$.

Conductor	Character	Characteristic polynomial	Result	Rank bounds
31	χ_{31}	$x^3 + x^2 - 10x - 8$	Verified	$[2,4]$
67	χ_{67}	$x^3 + x^2 - 22x + 5$	Verified	$[2,2]$
133	$\chi_7\chi_{19}$	$x^3 - x^2 - 44x + 64$	Verified	$[1,4]$
151	χ_{151}	$x^3 + x^2 - 50x - 123$	Lower bound too loose	$[0,2]$
193	χ_{193}	$x^3 + x^2 - 64x + 143$	Lower bound too loose	$[0,2]$
247	$\chi_{13}^2\chi_{19}$	$x^3 - x^2 - 82x + 64$	Lower bound too loose	$[0,2]$
469	$\chi_7\chi_{67}$	$x^3 - x^2 - 156x - 608$	Lower bound too loose	$[0,4]$
547	χ_{547}	$x^3 + x^2 - 182x - 81$	Lower bound too loose	$[0,4]$
589	$\chi_{19}^2\chi_{31}$	$x^3 - x^2 - 196x + 349$	Lower bound too loose	$[0,4]$
613	χ_{613}	$x^3 + x^2 - 204x + 999$	Lower bound too loose	$[0,2]$
679	$\chi_7\chi_{97}$	$x^3 - x^2 - 226x + 176$	Lower bound too loose	$[0,2]$
691	χ_{691}	$x^3 + x^2 - 230x + 128$	Lower bound too loose	$[0,4]$
703	$\chi_{19}\chi_{37}$	$x^3 - x^2 - 234x + 729$	Lower bound too loose	$[0,4]$
739	χ_{739}	$x^3 + x^2 - 246x - 520$	Lower bound too loose	$[0,4]$
817	$\chi_{19}^2\chi_{43}$	$x^3 - x^2 - 272x + 1755$	Lower bound too loose	$[0,4]$
853	χ_{853}	$x^3 + x^2 - 284x + 1011$	Lower bound too loose	$[0,2]$
871	$\chi_{13}^2\chi_{67}$	$x^3 - x^2 - 290x + 1000$	Lower bound too loose	$[0,2]$
1009	χ_{1009}	$x^3 + x^2 - 336x - 1719$	Lower bound too loose	$[0,4]$

E37A

Table 7: Results using Magma to perform descents on $E37A$.

Conductor	Character	Characteristic polynomial	Result	Rank bounds
43	χ_{43}	$x^3 + x^2 - 14x + 8$	Verified	$[3,3]$
61	χ_{61}	$x^3 + x^2 - 20x - 9$	Verified	$[3,3]$
103	χ_{103}	$x^3 + x^2 - 34x - 61$	Verified	$[3,3]$
127	χ_{127}	$x^3 + x^2 - 42x + 80$	Verified	$[3,3]$
171	$\chi_9\chi_{19}$	$x^3 - 57x - 19$	Verified	$[3,3]$
247	$\chi_{13}^2\chi_{19}$	$x^3 - x^2 - 82x + 64$	Verified	$[3,3]$
817	$\chi_{19}^2\chi_{43}$	$x^3 - x^2 - 272x + 1755$	Verified	$[3,3]$
853	χ_{853}	$x^3 + x^2 - 284x + 1011$	Verified	$[3,3]$

E40A3

Table 8: Results using Magma to perform descents on $E40A3$.

Conductor	Character	Characteristic polynomial	Result	Rank bounds
7	χ_7	x^3+x^2-2x-1	Verified	$[2,2]$
63	$\chi_9\chi_7$	$x^3-21x-28$	Verified	$[2,2]$
91	$\chi_7\chi_{13}$	$x^3-x^2-30x+64$	Verified	$[2,2]$
223	χ_{223}	$x^3+x^2-74x-256$	Verified	$[2,2]$
259	$\chi_7\chi_{37}$	$x^3-x^2-86x-48$	Verified	$[2,2]$
277	χ_{277}	$x^3+x^2-92x+236$	Lower bound too loose	$[0,2]$
301	$\chi_7^2\chi_{43}$	$x^3-x^2-100x+379$	Lower bound too loose	$[0,2]$
427	$\chi_7\chi_{61}$	$x^3-x^2-142x+680$	Lower bound too loose	$[0,2]$
499	χ_{499}	$x^3+x^2-166x+536$	Verified	$[2,2]$
547	χ_{547}	$x^3+x^2-182x-81$	Verified	$[1,2]$
619	χ_{619}	$x^3+x^2-206x+321$	Lower bound too loose	$[0,2]$
679	$\chi_7^2\chi_{97}$	$x^3-x^2-226x-503$	Lower bound too loose	$[0,2]$
757	χ_{757}	$x^3+x^2-252x+729$	Lower bound too loose	$[0,2]$
853	χ_{853}	$x^3+x^2-284x+1011$	Verified*	$[2,4]$
883	χ_{883}	$x^3+x^2-294x+1439$	Lower bound too loose	$[0,2]$
919	χ_{919}	$x^3+x^2-306x-1872$	Lower bound too loose	$[0,2]$
973	$\chi_7\chi_{139}$	$x^3-x^2-324x+36$	Lower bound too loose	$[0,2]$
1009	χ_{1009}	$x^3+x^2-336x-1719$	Verified*	$[1,4]$
1129	χ_{1129}	$x^3+x^2-376x-2927$	Lower bound too loose	$[0,2]$
1197	$\chi_9\chi_7^2\chi_{19}$	$x^3-399x+2926$	Lower bound too loose	$[0,2]$
1267	$\chi_7\chi_{181}$	$x^3-x^2-422x+3144$	Lower bound too loose	$[0,2]$
1359	$\chi_9\chi_{151}$	$x^3-453x+3473$	Lower bound too loose	$[0,2]$
1447	χ_{1447}	$x^3+x^2-482x+1715$	Lower bound too loose	$[0,2]$
1729	$\chi_7^2\chi_{13}\chi_{19}$	$x^3+x^2-576x-64$	Lower bound too loose	$[0,2]$

*: Unfortunately I forgot to record what the generators actually were for these cases!

2.2 Finding characteristic polynomials

This section discusses how to find the characteristic polynomial of the fixed field of χ given the information contained in the tables of $L(E,1,\chi)$ values. This is necessary because while the data in its original form is unwieldy for for the techniques being discussed.

For a fixed E the tables of values of $L(E,1,\chi)$ are given as a triplet $(k,r,L(E,1,\chi))$, where k is the conductor of χ and r is a positive integer which is used to further distinguish the characters when their conductor does not specify them uniquely.

The program findpolys.py works in two steps, first using (k,r) to construct a character $\chi_{k,r}:\mathbb{Z}/k\mathbb{Z}\to\mathbb{Q}(\zeta_3)$ explicitly, then finding the fixed field of that character.

2.2.1 (k,r) to $\chi_{k,r}$

It is first necessary to explain how r distinguishes between characters of the same conductor. An important convention is that k will always chosen to be minimal. A consequence of this is that k factors as $k=9^a p_1 p_2 \dots p_l$ with $p_i \equiv 1 \pmod 3$ for every i.

For conductor 9^a or p prime there are only three characters $\chi_p : \mathbb{Z}/p\mathbb{Z} \to \mathbb{Q}(\zeta_3)$. One is the trivial character and the other two are conjugates of one another. Because we assume k is minimal the trivial character is never the one a pair (k, r) will denote if $k > 1$. The pair of conjugate characters fix the same field so distinguishing between them is unnecessary. We construct the character χ_p by finding the least positive generator g of $(\mathbb{Z}/p\mathbb{Z})^*$ and set $\chi_p(g) = \zeta_3$, from which the rest of the character can be determined as the characters are multiplicative.

For composite $k = 9^a p_1 p_2 \dots p_l$ (with $p_i < p_{i+1}$ for every i) any character χ_k of conductor k factors as $\chi_k = \chi_{9^a}^{\epsilon_0+1}\chi_{p_1}^{\epsilon_1+1}\cdots\chi_{p_l}^{\epsilon_l+1}$ where ϵ_i is either 0 or 1 for every i. Consequently the characters of conductor k can be identified with an ordered set $(\epsilon_0, \epsilon_1, \dots, \epsilon_l)$. The conjugate of the character associated to $(\epsilon_0, \epsilon_1, \dots, \epsilon_l)$ is the character associated to $(1-\epsilon_0, 1-\epsilon_1, \dots, 1-\epsilon_l)$ (switching 0s and 1s). Since these fix the same field we don't need to distinguish between them, so by convention we impose that $\epsilon_l = 0$. The quantity r which appears in the L-function tables is $1 + \epsilon_0 2^0 + \epsilon_1 2^1 + \cdots + \epsilon_{l-1}2^{l-1}$, i.e. it is the string $\epsilon_0\epsilon_1\dots\epsilon_{l-1}$ interpreted as a binary number written backwards, plus 1.

Example: $\chi_{1953,2} = \chi_9^2\chi_7\chi_{13}$

First we factor 1953 as $9 \cdot 7 \cdot 13$, writing χ_{9^a} first if it appears and otherwise arranging the primes in increasing order. $r = 4$, so $r - 1 = 11_2$ (pad with zeros until $r-1$ has $l-1$ digits if necessary). This gives the ordered set of ϵ's as $(1, 1, 0)$ (remembering that by convention the last ϵ is always 0). We can then write $\chi_{1953,4} = \chi_9^{\epsilon_0+1}\chi_7^{\epsilon_1+1}\chi_{13} = \chi_9^2\chi_7^2\chi_{13}$.

As mentioned above, characters of prime conductor are produced by sending the least positive primitive root g of $(\mathbb{Z}/p\mathbb{Z})^*$ to $\zeta_3 = e^{2\pi i/3}$, and generating the rest of the character multiplicatively. Algorithmically,

```
def makechar(p): #generates \chi_p as an array of length p
  g = primitive_root(p); #gp function which returns least positive primitive root of Z/pZ^*
  zeta = exp(2*pi*i/3);
  char = [0]*p; #length p array of zeros
  for k in [1,p):
    char[g^k] = zeta^k;
  return char;
```

Having generated all the necessary χ_p's they are multiplied together to produce $\chi_{k,r}$. This is done as follows:

```
def muliplychars(list_of_characters, r): #list_of_characters are elements of the form (p, char_p)
  list_of_epsilons = makeepsilonlist(r); #use r as described above
  k = 1;
  for (p, char) in list_of_characters: #this loop reconstructs the conductor
    k = k*p;
  char = [1]*k; #list of ones of length k. This will be the \chi_{k,r}
  for i in [0,k):
    index = 0; #to index the primes with
    for (p, char_p) in list_of_characters:
      char[i] = char[i]*char_p[i%p]^list_of_epsilons(index); #i%p is i mod p
      index = index + 1;
  return char;
```

These two snippets of code are the essence of how findpolys.py generates $\chi_{k,r}$ from (k, r). There is of course a substantial amount of data management involved, but discussion of such is not within the scope of this document. Readers can reference findpolys_sage.py for this (and not findpolys.py, which uses gp for computations and is perhaps harder to follow because it loads other files). I have also not given any pseudocode for turning r into $(\epsilon_0, \dots)$. This is because languages will tend to have different ways to transform integers to binary. Users are best off referencing standard documentation for this. In python, the command bin(n) returns the string '0b<n in binary>' (for example bin(5) == '0b101').

2.2.2 Finding the fixed field of $\chi_{k,r}$

The key tool in producing the polynomial of the fixed field $K_{k,r}$ of $\chi_{k,r}$ is gp's function galoissubcyclo, which takes a subgroup H of $Gal(\mathbb{Q}(\zeta_n)/\mathbb{Q}) \cong (\mathbb{Z}/n\mathbb{Z})^*$ and returns the corresponding fixed subfield of $\mathbb{Q}(\zeta_n)$. galoissubcyclo can take many kinds of arguments, but the most convenient to pass in this context are n and a set of generators of H.

Producing all the elements in H can be done efficiently (in gp; sage is slow) with the naive approach of checking the elements of $(\mathbb{Z}/nZ)^*$ one at a time to see if they are fixed by $\chi_{k,r}$.

```
def find_fixed_elements(k, char):
  elements_of_H = [];
  for i in [0,k):
    if char[i] == 1:
      elements_of_H.append(i);
  return elements_of_H;
```

H will have $\frac{k-1}{3}$ elements, and k can be on the order of 10^5, so the array returned by the above snippet of code will typically be very large. Sage can pass this to gp as is without running into trouble and galoissubcyclo works as intended (since all the elements of H certainly form a set of generators of H), but for other applications it may be useful to find a smaller set of generators. Typically the values of k provided in the data have only a few prime factors, and hence sets of generators with only a few elements. Using a disjoint-set data structure and a union-find algorithm it is possible to reduce the size of the set of generators to only a few elements (roughly the number of prime factors of k), providing a portable representation of H.

Disjoint-set data structures are partitions of a given set of elements into disjoint subsets and union-find allows the following two operations to be performed:

Find Determine which subset a given element is in. In practice this is usually done by assigning a representative to each subset.

Union Perform the union of two subsets. In practice this might be implemented by setting the representative of every element of one set to the representative of the other.

Initially place every element of H into a singleton set containing only that element. The algorithm works in two steps. First, find an element g which is not in the same subset as 1. This will be a generator. Then, for every subset S union S with gS. This is equivalent to quotienting by g. Continue doing this until only one subset remains. The following pseudo-code is similar to the algorithm I've implemented.

```
def step1(set_system, k): #set system is the set of subsets
  for g in range [0,k):
    if set_system.representative[g] != set_system.representative[1]:
    #set_system.representateive[g] returns the representative of the set containing g
      return g;

def step2(set_system, g):
  set_of_representatives_copy = set_system.representative.values();
  #make a copy because usually looping over a changing set which is modified
  # in the loop is either problematic or explicitly disallowed
  for a in set_of_representatives_copy:
    set_system.union(a, g*a); #union the sets containing a and g*a

def make_small_set_of_generators(H, k):
  set_system = {} #a dictionary in python
  generators = [];
  for h in H:
    set_system.union(h,h); #initialize to every element in a singleton set
  while len(set_system.representative.values()) > 1: #while there is more than one subset
    g = step1(set_system, k);
    generators.append(g);
    step2(set_system, g);
  return generators;
```

Unfortunately this process is not guaranteed to produce a minimal set of generators, or even a superset of a minimal set of generators. For example, $\{2, 6\}$ is such a set in $(\mathbb{Z}/7\mathbb{Z})^*$, though it so happens that, because the program tries elements of H in order, if it were run on all of $(\mathbb{Z}/7\mathbb{Z})^*$ it would produce $\{2, 3\}$, which is a superset of a minimal set of generators, since 3 is a primitive root.

At this point k and a set of generators of H are passed to gp's galoissubcyclo, which returns a characteristic polynomial of the fixed field. About two thirds of the program's runtime is taken up by constructing the characters and finding its fixed values, and one third by galoissubcyclo. For $k \approx 3 \cdot 10^5$, findpolys takes about 6 minutes to run. Finding the polynomials of all the fields described in the data for a single curve takes a couple days.

2.3 Brute Force Search

The affine part of an elliptic curve E/K can be described by the locus satisfying the curve's Weierstrass equation, $E : F(x, y) = 0$ with $x, y \in K$, a polynomial equation in two variables. If x is fixed, the equation becomes a quadratic polynomial in only one variable, for which finding roots is straightforward. Brute Force Search (BFS) generates a set X_0 of values $x_0 \in K$ and then checks if the one variable polynomial $F(x_0, y)$ has any solutions for $y \in K$. For the purposes of determining whether or not $\text{rank}_{\mathbb{Z}} E(K) > \text{rank}_{\mathbb{Z}} E(\mathbb{Q})$ it is enough to find a single point of infinite order, but for other applications it may be useful to find as many points as possible. Code for both uses is included.

For a given cyclic cubic extension K of $\mathbb{Q}$ define α to be the generator of K. Any $x_0 \in K$ can be described by a triplet of rational numbers (p_0, p_1, p_2) by writing x_0 as $p_0 + p_1\alpha + p_2\alpha^2$, so to produce values x_0 it is enough to produce rationals. Any algorithm which produces rationals will likely manifest itself as a map $f : \mathbb{N} \to \mathbb{Q}$. Ideally this map should have the following properties for this purpose:

1. The map should be bijective. Injectivity is important because it avoids wasting time repeating the same computations, and surjectivity is important because it makes results easier to understand and reproduce, and is more likely to provide partial results.
2. The map should map small n to rationals of small height because, in accordance with the law of small numbers, rationals of small height are in some sense more likely to satisfy E than rationals of large height. It also helps the algorithm be easier to understand.
3. It should be possible to generate the image of f recursively. This is useful because it avoids having to store the entire image of f in memory. For use with BFS this criterion is unimportant, however, as any usually only hundreds, or at most thousands of rationals can be tested before runtimes become impractically long (weeks). This sequence is used for other purposes however, where this might be a concern.

The usual Calkin-Wilf sequence has the second and third properties, and is bijective with $\mathbb{Q}^+$, which can easily be extended to all of $\mathbb{Q}$. It is given by the recursion

$$q_{i+1} = \frac{1}{2\lfloor q_i \rfloor - q_i + 1}$$

with $q_0 = 1$. It begins $\frac{1}{1}, \frac{1}{2}, \frac{2}{1}, \frac{1}{3}, \frac{3}{2}, \frac{2}{3}, \frac{3}{1}, \frac{1}{4}, \frac{4}{3}, \frac{3}{5}, \frac{5}{2}, \frac{2}{5}, \frac{5}{3}, \frac{3}{4}, \ldots$. Setting $f(2n + 1) = q_n$, $f(0) = 0$, and $f(2n) = -f(2n - 1)$ for $n > 0$ (i.e. adding 0 to the beginning of the image and adding negatives of all the elements) gives a map which satisfies the three criteria above. Here is some code which generates this map:

```
def calkinwilf(depth): #generates depth elements
  cwseq = [0,0]; #include twice to not break the recursion
  for i in [0,depth/2):
    cwseq.append(1/(2*floor(cwseq[-2]) - cwseq[-2] + 1));
    #cwseq[-2] is the second from last element of the sequence
    cwseq.append(-cwseq[-1]);
  del cwseq[0]; #remove the first 0.
  return cwseq;
```

Instead of appending the elements to a list to be returned one could just as easily perform computations with them.

The bare bones of BFS is as described above: pick values $x_0 \in K$ and see if $F(x_0, y)$, as a polynomial in y, has any roots. In code:

```
for p_0 in cwseq: #cwseq is a sequence of rationals
  for p_1 in cwseq:
    for p_2 in cwseq:
      x_0 = p_0 + p_1*a + p_2*a^2; #a is the generator of K
      F = y^2 + a_1*x_0*y + a_3*y^2 - (x_0^3 + a_2*x_0^2 + a_4*x_0 + a_6); #F(x_0,y)
      roots_of_F = F.roots();
      if length(roots_of_F) > 0: #if F(x_0,y) has roots in K
        for y_0 in roots_of_F:
          if (x_0, y_0) not in Q^2:
            return (x_0, y_0);
```

There are various improvements that can be made to this algorithm but the above is the essence of it.

One modification that can be made is to make use of the group law on E to find additional points and to attempt to determine a minimal generating set of points. I implement a method to do this which works in practice but will fail in general.

During the search, whenever a point P is found I check to see if it has already been found or generated by the procedure described below. If not. nP and $-mP$ are included in the set of all points discovered, with certain stopping conditions on n and m. The sets $\{nP\}$ and $\{-mP\}$ are treated completely separately. n begins at 1 and is incremented until either $nP = \mathcal{O}$ or the denominator of $x(nP)$ exceeds some fixed H_0 which can be safely set to about 10^{10}. m is treated in the same way. The reasoning is that in the first case P has finite order, so all of P's multiples are easily included. In the second case Nagell-Lutz and the generalization to arbitrary number fields allows us to deduce that P has infinite order. The maximum size of the denominator of the x coordinate will depend on the number field, but for the cases we are interested in it will be less than 10^{10}. Setting H_0 to be large does not cause problems with runtime or memory because the (logarithmic) height of $2P$ is roughly 4 times the height of P, so even for moderate n the height of nP will greatly exceed any reasonable bounds we could place.

If P is found to have infinite order in this way P is added to a set of "potential generators" and any multiples of P which are in this set are removed. After having checked all the x values that are to be checked, as above, the program ends up with a set of points found and a set of potential generators. The program then computes the span of all subsets of the potential generators from smallest to largest, and returns the first subset for which the span contains all the potential generators. In my experience this has never failed, but it will not work in general. First, it assumes that there are no generators with denominator larger than H_0. This objection is moot in practice because any reasonable search will not find any points with denominator larger than H_0. Second, it is impossible to generate the entire span of subsets of potential generators because these spans are infinite. Instead the program only generates linear combinations with coefficients less than a certain fixed bound, which I've set in the past to be roughly 20. If there's a "complicated" linear dependency the program will not find it. Third, it may be that, as mentioned towards the end of section

2.2.2, the potential generators may not contain a minimal set of generators at all. Also note that if a point is found, it's galois conjugates will also be points. The program as is does not take advantage of this fact though.

Though for this application it is unnecessary to find as many points as possible, it's conceivable that for other applications it would be desirable. Code using the technique implemented above can be found in bruteforcesearch_manypoints.py.

Another improvement that can be made is to iterate over triples (p_0, p_1, p_2) in a more natural way. The algorithm described above fixes p_0 and p_1 before iterating over all possible values of p_2, and so $(0, 0, \frac{8}{7})$ is tested before $(1, 0, 0)$. By finding a bijection $g_{1,3} : \mathbb{N} \to \mathbb{N}^3$ which maps small naturals to triplets of small naturals one can improve the efficiency of the search. One such map is given in ntoncubed.py.

It may seem as though this consideration is superfluous given that the majority of searches find no points at all, making it unnecessary to optimize the order in which the points are checked in all but a few cases. The value of modifying the search order to examine "small" triplets before large ones is that small ones are more likely to satisfy equations in some sense, and with this procedure the search as a whole is run more efficiently because it is examining the x_0 values most likely to satisfy E. For example, with the bijection given in ntoncubed.py, $\{g(k)\}_{k=0}^{219}$ is the set of all triplets with entries that sum to 9 or less (and this is optimal), whereas proceeding by cubes as above requires $0 \leq k < 1000$. In the limit this ratio approaches $\frac{1}{6}$.

However I have not made use of this improved order to examine triplets in the computations I've run because it is less economical in terms of memory, which is a serious concern given that I am sharing the Université de Montréal server with several others. It would likely not be difficult to modify the construction of the image of g in a way that only a small amount of memory is used at any one time.

Given how ineffective this technique is, however, as evidenced by the results above, time is probably better spent elsewhere.

2.4 Descent with Magma

Magma is a Computer Algebra System developed at the University of Sydney. It has many powerful high level computational tools, and in particular it can perform descent using a function called MordellWeilShaInformation. The documentation, found at http://magma.maths.usyd.edu.au/magma/handbook/text/1353#14971, reads as follows:

> This is a special function which uses all relevant Magma machinery to obtain as much information as possible about the Mordell-Weil group and the Tate-Shafarevich group of the elliptic curve E. The tools used include 2-descent and the Cassels-Tate pairing on the 2-Selmer group; analytic routines may also be used when the conductor has small norm.

In light of this the second technique consists of simply running MordellWeilShaInformation on all the curves of interest. The code I use to do this can be found in the files findrank¡curvename¿.magma. I don't have very much to say about this method because Magma does not offer any options to customize how the descent is performed as far as I can tell. Moreover I've only been able to use this technique on four curves because of the limitations on l'Université de Montréal computational resources, so it is difficult for me to make more general statements about how this technique performs in practice.

This approach has been the most effective of the three I've tried. Its effectiveness seems to depend on whether or not the curve in question has two torsion. $E37A1$ does not, and descent there takes a very long time: E/K_{853} took the better part of a week to return, and the runtime seems to grow much faster than linearly in the conductor. However it seems, based on only 8 results, to be successful more often than descent for the other three curves, which all have two torsion. The descent with the other three curves tended to run quickly, often taking only minutes per result, though sometimes Magma would print a warning that a large number (≈ 150 digits) was being factored, and such instances made up the largest portion of the runtime.

2.5 Sieving with projections

This technique is essentially a refinement of brute force search where, instead of setting (using the same notation as above) $X_0 = K$, i.e. checking for every $x_0 \in K$ whether or not $F(x_0, y) = 0$ has a solution, local information is used to produce a much smaller set $X_S \subset X_0$ while still containing all x_0 values for which $F(x_0, y) = 0$ has a solution. This is done by reducing the equation modulo an ideal $I \subset \mathcal{O}_K$, the ring of integers of K, finding the set of all solutions $\overline{X}_S$ to the reduced equation, and then taking X_S to be the preimage of $\overline{X}_S$ in K. However I was not able to find an efficient way of determining the preimage; my implementation of this idea was roughly on par with brute force search.

2.5.1 Finding $\overline{X}_S$

The current implementation of this algorithm is naive. Given an ideal $I \subset \mathcal{O}_K$, the program checks for each $\bar{x}, \bar{y} \in \mathcal{O}_K/I$ whether or not $\overline{F}(\bar{x}, \bar{y}) = 0$.

```
def solutions_mod_I(E, K, I):
  O_KmodI = K.ring_of_integers().quotient_ring(I);
  Xbar_S = [];
  for x in O_KmodI:
    for y in O_KmodI:
      Fxy = y^2 + a1*x*y + a3*y - (x^3 + a2*x^2 + a4*x + a6);
      if Fxy = 0:
        Xbar_S.append(x);
  return Xbar_S;
```

This algorithm could be improved by first finding the prime ideals of $\mathcal{O}_K$ and taking I to be one of them. The quotient $\mathcal{O}_K/I$ would then be a finite field and more root finding techniques would become available. If the preimage of $\overline{X}_S$ could be determined efficiently, this modification, combined with a suggestion below to use the Chinese Remainder Theorem (CRT) to turn several small sieves into one larger one, could potentially improve the algorithm substantially. Of course I have no evidence to support this claim, but I give a heuristic justification below.

2.5.2 Computing the preimage of $\overline{X}_S$

Given some $x = \frac{p_0}{q_0} + \frac{p_1}{q_1}\alpha + \frac{p_2}{q_2}\alpha^2 \in K$ and some ideal $I \subset \mathcal{O}_K$, if q_0, q_1, and q_2 have inverses in $\mathcal{O}_K/I$ then there is a projection $\pi : K \to \mathcal{O}_K/I$ with $\pi(x) = \bar{p_0}\bar{q_0}^{-1} + \bar{p_1}\bar{q_1}^{-1}\bar{\alpha} + \bar{p_2}\bar{q_2}^{-1}\bar{\alpha}^2$, where $\bar{\cdot} : \mathcal{O}_K \to \mathcal{O}_K/I$ is reduction modulo I. The step above computes the set $\overline{X}_S \subseteq \mathcal{O}_K/I$ of all the residues which solved the reduced equation $\overline{F}(\bar{x}, \bar{y}) = 0$, and now we seek to determine $X_S = \pi^{-1}(\overline{X}_S)$. This is done in two steps.

Pick representatives in $\mathcal{O}_K$ for every coset in $\mathcal{O}_K/I$. It will be useful to formalize this idea with the use of a map $\psi : \mathcal{O}_K/I \to \mathcal{O}_K$ which maps a coset to its representative. For each $\bar{x}_S \in \overline{X}_S$, the set

$$P(\bar{x}_S) = \left\{ \frac{\psi\left(\overline{\psi(\bar{x}_S)\psi(g)}\right)}{\psi(g)} \right\}_{g \in (\mathcal{O}_K/I)^*}$$

is the first ingredient in the construction of X_S. The notation above is maybe a bit verbose. The nested maps are to ensure that the computations are being performed in the right ring, but they are all either lifts or projections. The key point is that the product in the numerator is done in $\mathcal{O}_K$ and then the corresponding representative is chosen for the numerator of the field element.

$P(\bar{x}_S)$ is the (complete) preimage of $\overline{X}_S$ if restricted to only to the representatives $\psi(\mathcal{O}_K/I)$. For example, if $K = \mathbb{Q}$, $I = 5\mathbb{Z}$, and our set of representatives is $\{0, 1, 2, 3, 4\}$ we have $P(2) = \{2, \frac{1}{3}, \frac{3}{4}\}$, which is the complete preimage of 2 mod 5 in the subset of $\mathbb{Q}$ with numerators and denominators in the set of representatives, as expected.

One way to compute $P(\bar{x}_S)$ for a given $\bar{x}_S$ is shown below.

```
def P(K, I, x):
  O_K = K.ring_of_integers();
  O_KmodI = O_K.quotient_ring(I);
  Px = [];
  for g in O_MmodI.unit_group():
    p = lift((lift(g)*lift(x)).mod(I))/lift(g);
    Px.append(p);
  return Px;
```

To generate the rest of the preimage X_S we first produce $P(\overline{X}_S) = \bigcup P(\bar{x}_S)$. From there all that will be left to do to obtain all of X_S is to shift both the numerators and denominators of $P(\overline{X}_S)$ by elements of I.

The motivation behind the shifting is that whenever ψ is applied to an element of $\mathcal{O}_K/I$ a certain generality is lost through our choice of representatives. To recover the entire preimage of $\overline{X}_S$ we must shift these outputs by every element of I. It's unnecessary to do this for the argument of the outermost calling of ψ in the numerators of the elements of $P(\bar{x}_S)$ however, as immediately afterwards the product will be reduced once again and any shifts will vanish. In light of this we can deduce that the entire preimage is given by

$$X_S = \left\{ \frac{p+n}{q+m} \;\middle|\; \frac{p}{q} \in P(\overline{X}_S), n, m \in I \right\}$$

where $\frac{p}{q}$ is taken to be of the form given in the definition of $P(\bar{x}_S)$. Given that we want to preserve the specific form of the elements of $P(\overline{X}_S)$ it is safest to modify the algorithm for P above so that it instead returns an ordered pair.

Of course in practice it will not be possible to do this for every $n, m \in I$. One way to approximate this is to find maps $f : \mathbb{N} \to \mathbb{N}^d$ (where d is the number of generators of I) and $g : \mathbb{N} \to \mathcal{O}_K$, apply g to every coordinate of the output of f, and take $n \in I$ to be $(g(f_1), g(f_2), \ldots, g(f_d)) \cdot (\beta_1, \beta_2, \ldots, \beta_d)$, where $f(n) = (f_1, f_2, \ldots, f_d)$, $\{\beta_i\}$ are a set of generators for I, and $\cdot$ is the dot product. As usual it's best if these maps map "simple" elements to simple elements.

One final consideration is that there will be elements which will cannot be projected onto $\mathcal{O}_K/I$, so X_S as is will not contain them. This is easily accounted for: one must include in P all elements of the form $\frac{r}{s}$ with $r \in \psi(\mathcal{O}_K/I)$ and $s \in \psi(\mathcal{O}_K/I - (\mathcal{O}_K/I)^*)$. A complete algorithm to compute the preimage might look like

```
def find_preimage(K, I, Xbar_S, depth): #depth is the number of pairs (n,m)
  O_K = K.ring_of_integers();
  Pvals = []; #going to be pairs (x, P(x)) = (x, (p,q))
  for x in Xbar_S:
    Px = P(K, I, x);
    Pvals.append((x, Px));
  Igens = I.generators();
  d = length(Igens);
  fimage = f(d, depth); #generates the first depth elements of N -> N^d
  gimage = g(O_K, depth)
  #really only about sqrt(depth) are needed
  NtoN2 = ntonsquared(depth);
```

```
X_S = [];
for (x, (p,q)) in Pvals:
  for i in [0,depth):
    (j,k) = NtoN2[i];
    nindices = fimage[j]; #numbers to feed to g
    mindices = fimage[k];
    ncoeffs = [];
    for nindex in nindices:
      ncoeffs.append(gimage[nindex]);
    for mindex in mindices:
      mcoeffs.append(gimage[mindex]);
    n = dot(ncoeffs, Igens); #dot product
    m = dot(mcoeffs, Igens);
    x = (p + n)/(q + m);
    X_S.append(x);
return X_S;
```

The algorithm then proceeds to run brute force search as described above on X_S. In practice it is best if $x \in X_S$ is tested as it is created instead of first generating a list and then running the search on it since otherwise millions of number field elements would have to be kept in memory. Instead of appending x to X_S above simply test immediately. In practice it will also likely be necessary to catch division by zero errors.

2.5.3 Improving and the Chinese Remainder Theorem

The motivation for using X_S as notation is that S can designate an ideal, or a set of ideals. Using CRT it is possible to combine the sets X_{S_1} and X_{S_2} into $X_{S_1 \cup S_2}$. This is interesting because combining the two sets like this will be substantially faster than running the algorithm on $X_{S_1 \cup S_2}$ directly, since, very roughly, the run time for the former's runtime will grow with the sum of the sizes of the ideals in $S_1 \cup S_2$, since it amounts essentially to determining X_S for each of them, while the latter's will grow with the product of the sizes, since the algorithm above will check every residue in $\mathcal{O}_K/I_1 I_2 \dots$. Moreover it's advantageous to look at situations where $I_1 I_2 \dots$ is large, since if the acceptable residues of $\mathcal{O}_K/I_1$ and $\mathcal{O}_K/I_2$ are independently distributed the number of acceptable residues will drop exponentially with the number of ideals in the product.

However I had difficulty computing X_S efficiently to begin with and never attempted to go beyond that hurdle to implement the CRT technique mentioned above. However, unbeknown to me at the time I worked on this program Michael Stoll has written a program called ratpoints[2] which finds rational solutions to equations of the form $y^2 = p(x)$ using essentially this technique. It is effective, which makes me hopeful that with a proper treatment this too could be a viable technique for finding points on these curves.

3 Curve Finding

Another approach to be taken for verifying BSD is to construct curves with positive rank and check to see if $L(E, 1) = 0$. Here we focus again on curves over cubic number fields, and are interested in finding, for a given E, cubic number fields K such that $\mathrm{rank}_{\mathbb{Z}} E(K) > \mathrm{rank}_{\mathbb{Z}} E(\mathbb{Q})$. A cubic extension of $\mathbb{Q}$ is Galois ("cyclic") if and only if the discriminant of its characteristic polynomial is square.[3] We can think of (the affine part of) an elliptic curve as the locus in which solves $E : x^3 + a_2 x^2 + (a_4 - a_1 y)x + (a_6 - a_3 y - y^2) = 0$.

[2]Which can be found at http://www.faculty.jacobs-university.de/mstoll/programs/index.html.

[3]This is a well known result as far as I can tell, since no one seems to give a reference for it, but see Harvey Cohn's *The density of abelian cubic fields*, Proc. Amer. Math. Soc. 5 (1954), 476-477, for example.

Fixing y allows $E = E_y(x)$ to be considered as a univariate polynomial, and if $\mathbb{Q}[x]/(E_y(x)) \cong \mathbb{Q}[\alpha] \overset{\text{def}}{=} K$ is a cyclic cubic extension of $\mathbb{Q}$ with $E_y(x)$ as a characteristic polynomial then clearly $(\alpha, y) \in E(K)\backslash E(\mathbb{Q})$, so either $\text{rank}_{\mathbb{Z}}E(K) > \text{rank}_{\mathbb{Z}}E(\mathbb{Q})$ or $|E(K)| > |E(\mathbb{Q})|$. Often the former will be the case, giving us what we're looking for. We can then compute $L(E, \chi, 1)$ to check if it is 0. The difficulty lies in picking y such that the discriminant $\Delta(E_y)$ is a perfect square in $\mathbb{Q}$. The code in findcyclicextensions.py can find cyclic extensions with a good success rate using the procedure described below. I feel that any pseudo-code will be too complex to be useful, but readers can always consult the python file if they like.

The discriminant $\Delta(E_y)$ is a quartic polynomial $\alpha y^4 + \beta y^3 + \gamma y^2 + \delta y + \epsilon$ with

$$\begin{aligned}
\alpha &= -27\\
\beta &= 4a_1^3 - 54a_3 + 18a_1a_2\\
\gamma &= a_1^2a_2^2 - 12a_1^2a_4 + 4a_2^2 - 27a_3^2 + 54a_6 - 18a_2a_4 + 18a_1a_2a_3\\
\delta &= -2a_1a_2^2a_4 + 12a_1a_4 + 4a_2^3a_3 + 54a_3a_6 - 18a_2a_3a_4 - 18a_1a_2a_6\\
\epsilon &= a_2^2a_4^2 - 4a_4^3 - 4a_2^3a_6 - 27a_6^2 + 18a_2a_4a_6
\end{aligned}$$

Consider the curve $C_\Delta : v^2 = \alpha u^4 + \beta u^3 + \gamma u^2 + \delta u + \epsilon$. Points (u, v) on this curve correspond to pairs $(y, \Delta(E_y))$ which yield square discriminants and hence cyclic cubic extensions of $\mathbb{Q}$ over which either $\text{rank}_{\mathbb{Z}}E(K) > \text{rank}_{\mathbb{Z}}E(\mathbb{Q})$ or $|E(K)| > |E(\mathbb{Q})|$. This curve, the *quartic discriminant curve*, can be turned into an elliptic curve if a single rational point (p, q) can be found on it. Michael Stoll's ratpoints program, mentioned in section 2.5.3, often does this effectively. It is possible that no points can be found like this. A partial solution to this problem is discussed below.

Once a point (p, q) on C_Δ is found it is then possible to transform the curve into Weierstrass form. There are two cases to consider when transforming.

Case 1: $q = 0$

In this case we can write $C_\Delta : v^2 = (u - p)(au^3 + bu^2 + cu + d)$. The transformation from variables (u, v) to (X, Y) given by

$$u = \tfrac{A}{X} + p$$

$$v = \tfrac{Y}{A}(u - p)^2$$

with $A = ap^3 + bp^2 + cp + d$ turns C_Δ to an elliptic curve in Weierstrass form with coefficients

$$\begin{aligned}
a_1 &= 0\\
a_2 &= 3ap^2 + 2bp + c\\
a_3 &= 0\\
a_4 &= A(3ap + b)\\
a_6 &= A^2a
\end{aligned}$$

Case 2: $q \neq 0$

First set $\bar{u} = u - p$. This change of coordinates turns C_Δ into $C_D : v^2 = a\bar{u}^4 + b\bar{u}^3 + c\bar{u}^2 + d\bar{u} + q^2$, where

$$\begin{aligned}
a &= \alpha\\
b &= 4\alpha p + \beta\\
c &= 10\alpha p^2 + 3\beta p + \gamma\\
d &= 4\alpha p^3 + 3\beta p^2 + 2\gamma p + \delta
\end{aligned}$$

Setting

$$\bar{u} = \frac{2q(X+c) - \frac{d^2}{2q}}{Y}$$

$$v = \frac{\bar{u}(\bar{u}X - d)}{2q} - q$$

transforms C_D into the equation for an elliptic curve in Weierstrass form with
$a_1 = \frac{d}{q}$
$a_2 = c - \frac{d^2}{4q^2}$
$a_3 = 2qb$
$a_4 = -4q^2a$
$a_6 = a_2a_4 = ad^2 - 4q^2ac$

It is important to distinguish between the quartic discriminant curve and this curve even if from a more theoretical perspective they coincide. The Weierstrass form of the quartic discriminant curve I will call the *cubic discriminant curve with distinguished point* (p, q) and denote it $E_\Delta(p, q)$ or simply E_Δ when the choice of distinguished point can be omitted. All the cubic discriminant curves related to a quartic discriminant curve are isomorphic, i.e. in some sense the choice of base point doesn't matter, but once again for computational applications it is often important to distinguish between them.

Obtaining the cubic discriminant curve is useful because there are many powerful computational tools available to determine the generators of $E_\Delta(\mathbb{Q})$. If E_Δ has positive rank and a generator can be found then infinitely many cyclic cubic extensions K for which $\text{rank}_\mathbb{Z}E(K) > \text{rank}_\mathbb{Z}E(\mathbb{Q})$ or $|E(K)| > |E(\mathbb{Q})|$ can be found: for each $(X, Y) \in E_\Delta(\mathbb{Q})$ there is a corresponding (u, v) on the quartic discriminant curve which gives a y such that $\Delta(E_y)$ is a square.

Given the generators $g_1, g_2, \dots, g_k$ of E_Δ I construct a set of points G by combining generators in more and more "complicated" ways, and then shift G by every torsion point successively. G is constructed by first including $g_1, -g_1, g_2, -g_2, \dots, g_k, -g_k$, then $2g_1, -2g_1, g_1 + g_2, g_1 - g_2, \dots$, i.e all two term sums of the generators and their negatives, and then all three term combinations and so on. Some of these will overlap, e.g. $g_1 + g_2 - g_1$ and g_1, but weeding after every iteration is quick enough. Call T the set of torsion points. Once G is constructed, the program then computes $T + G = \{t + g \mid t \in T, g \in G\}$, and returns this set of points when it is large enough to satisfy the user.

This approach is quick in many cases but could fail in two places. The algorithm may not be able to find any points on the quartic discriminant curve, or it might take an unreasonable amount of time to find the generators of E_Δ. Both of these problems can be circumvented to some extent by applying the procedure above not just for E as is, but also for E under the transformation $y \to y + rx + s$. This is the most general transform that will preserve the Weierstrass form while changing the discriminant. It affects the coefficients of E in the following way:
$a_1' = a_1 + 2r$
$a_2' = a_2 - r^2 - a_1r$
$a_3' = a_3 + 2s$
$a_4' = a_4 - 2rs - a_1s - a_3r$
$a_6' = a_6 - s^2 - a_3r$

I call the resulting curves *shears* of E. From a more abstract perspective E and its shears coincide, but for these more computational purposes the distinction is important, since there are many cases where a point cannot be found on the quartic discriminant curve of E but where one can be found on the quartic discriminant curve of a shear E' of E. Similarly maybe finding the generators of the cubic discriminant curve of E will take too long, but the generators of the cubic discriminant curve of E' can be found quickly. My implementation of this idea does the former but not the latter since right now the code serves as a proof of concept more than anything else.

The biggest problem with this technique is that the fields which are generated tend to have very large conductor, making finding $L(E, \chi, 1)$ essentially impossible. This is because adding points tends to increase their height substantially, so the coefficients of the characteristic polynomial can easily be hundreds of digits long. A way to mitigate this somewhat is to simply search for points on the quartic discriminant curves many shears of E using ratpoints.

To use this technique one also needs a way to find the character χ which fixes any fields produced by this technique. Presumably calculating the field's conductor shouldn't be hard, but I have not had time to think about how to distinguish between the characters of a given conductor. Depending on the conductor's size and the number of factors it has it may be feasible to put the characteristic polynomials produced into some canonical form and compare them with the canonical forms of the fixed fields of the characters of the right conductor.

Once χ has been identified the algorithm from prof. Kisilevsky and prof. Fearnley can be used to compute $L(E, \chi, 1)$. I expect that this approach will be more successful than trying to find points explicitly, but it may not be as useful because it restricts itself to fields of a certain form it seems.

4 Concluding Remarks

I employed three techniques to find points on specific elliptic curves. The naive search was unsuccessful in all but the simplest cases. Using Magma to perform descent worked a substantial fraction of the time, but the computations were lengthy. A third technique which was a refinement of the naive search was unsuccessful because I was not able to perform certain steps efficiently, but the technique may be viable with a better implementation.

I also outline a technique to find cyclic extensions of $\mathbb{Q}$ over which a given elliptic curve will have either more points or a higher rank. If one can find the character which fixes the resulting number fields one could compute the relevant L-functions as prof. Kisilevsky and prof. Fearnley do, and verify the Birch and Swinnerton-Dyer conjecture that way.

Finally I'd like to thank prof. Henri Darmon with utmost sincerity for guiding me through my research. His insight and expertise were invaluable to me.

Appendix: Generators found with Magma

This section consists of tables of the generators found with Magma using its descent algorithm. They are listed here and not above because of typesetting concerns. Only the cases where the algorithm was successful in finding generators are given here. For a complete summary of the results see the results section.

E20A2

Table 9: Results using Magma to perform descents on $E20A2$.

Conductor	Character	Generators
9	χ_9	$(-\alpha^2+2, \alpha^2-3)$, $(-\alpha+1, -\alpha^2+\alpha)$
63	$\chi_9\chi_7$	$(\frac{1}{7}\alpha, \frac{1}{7}(-\alpha+2))$, $(\frac{1}{12}(\alpha^2-4\alpha-9), \frac{1}{24}(-3\alpha^2+12\alpha+31))$
73	χ_{73}	$(\frac{1}{7803}(-134\alpha^2+766\alpha+6573), \frac{1}{397953}(14260\alpha^2+24232\alpha-145239))$, $(\frac{1}{1323}(-101\alpha^2-65\alpha+2028), \frac{1}{27783}(-2929\alpha^2-1885\alpha+57489))$
91	$\chi_7^2\chi_{13}$	$(\frac{1}{2187}(298\alpha^2-2308\alpha-873), \frac{1}{59049}(-29696\alpha^2+161948\alpha+155673))$, $(\frac{1}{2541}(-160\alpha^2+598\alpha+2319), \frac{1}{27951}(-640\alpha^2+2392\alpha-525))$
117	$\chi_9^2\chi_{13}$	$(\frac{1}{484}(53\alpha^2-329\alpha+94), \frac{1}{5324}(1855\alpha^2-11515\alpha+4258))$, $(\frac{1}{484}(-\alpha^2-211\alpha+530), \frac{1}{5324}(-449\alpha^2+3029\alpha-4998))$
133	$\chi_7^2\chi_{19}$	$(18\alpha^2+140\alpha+249, \frac{1}{3}(2896\alpha^2+20068\alpha+27723))$, $(\frac{1}{513}(-716\alpha^2+1978\alpha+28011), \frac{1}{1539}(-14320\alpha^2+39560\alpha+560193))$
171	$\chi_9^2\chi_{19}$	$(\frac{1}{12482}(-137\alpha^2-1771\alpha+6326), \frac{1}{986078}(12193\alpha^2+157619\alpha-363302))$, $(\frac{1}{256}(-9\alpha^2-280\alpha+1473), \frac{1}{4096}(-553\alpha^2+12008\alpha-44319))$
229	χ_{229}	$(\frac{1}{23762}(507\alpha^2-3847\alpha-38068), \frac{1}{2590058}(-40673\alpha^2-140939\alpha+2518600))$, $(\frac{1}{11881}(164\alpha^2+850\alpha-13545), \frac{1}{1295029}(43038\alpha^2-124384\alpha-1990771))$
259	$\chi_7\chi_{37}$	$(\frac{1}{24}(-41\alpha^2+67\alpha+3504), \frac{1}{48}(-991\alpha^2+1541\alpha+84312))$, $(\frac{1}{63}(-2\alpha^2+5\alpha+151), \frac{1}{189}(10\alpha^2-25\alpha-773))$
277	χ_{277}	$(\frac{1}{2768896}(-3451\alpha^2-93640\alpha+4134204), \frac{1}{4607442944}(100471719\alpha^2-576955672\alpha-7383525164))$, $(\frac{1}{87616}(-28309\alpha^2-111727\alpha+2349378), \frac{1}{12967168}(-20236113\alpha^2-79467595\alpha+1620176362))$
307	χ_{307}	$(\frac{1}{32}(\alpha^2-11\alpha-34), \frac{1}{384}(31\alpha^2-245\alpha-510))$, $(\frac{1}{16}(-\alpha^2+2\alpha+88), \frac{1}{192}(-23\alpha^2+22\alpha+2424))$
703	$\chi_{19}\chi_{37}$	$(\frac{1}{4}(\alpha^2+8\alpha+16), \frac{1}{8}(23\alpha^2+184\alpha-1217))$, $(\frac{1}{77841}(-542\alpha^2-4032\alpha+66663), \frac{1}{21717639}(75880\alpha^2+564480\alpha-2404971))$, $(\frac{1}{38475}(494\alpha^2-8846\alpha+23247), \frac{1}{577125}(21736\alpha^2-389224\alpha+1000593))$, $(\frac{1}{4}(-\alpha^2+\alpha+331), \frac{1}{8}(23\alpha^2-23\alpha-6028))$
727	χ_{727}	$(\frac{1}{1000}(-29\alpha^2+63\alpha+6012), \frac{1}{10000}(-667\alpha^2+1449\alpha+155276))$, $(\frac{1}{34680}(-211\alpha^2+1617\alpha+24028), \frac{1}{235824}(633\alpha^2-4851\alpha+1900))$, $(\frac{1}{15870}(-113\alpha^2-19\alpha+23454), \frac{1}{365010}(3729\alpha^2+627\alpha-721082))$, $(\frac{1}{4369210}(-23139\alpha^2+47463\alpha+2818762), \frac{1}{2888047810}(-9972909\alpha^2+20456553\alpha-139568678))$
763	$\chi_7^2\chi_{109}$	$(\frac{1}{1338248}(8999\alpha^2+91031\alpha-1420222), \frac{1}{1094686864}(7608739\alpha^2+41326507\alpha-1308282062))$
829	χ_{829}	$(\frac{1}{4044121}(12582\alpha^2+26568\alpha-98361), \frac{1}{8132727331}(1887024\alpha^2+314114892\alpha+1108142463))$, $(\frac{1}{44485331}(-89478\alpha^2+985554\alpha+40392189), \frac{1}{89460000641}(-323550012\alpha^2-1712970168\alpha+74853491325))$

Continued on next page

E20A2 Descent with Magma results – continued from previous page

Conductor	Character	Generators
1027	$\chi_{13}^2\chi_{79}$	$(\frac{1}{1757007}(2684\alpha^2-67138\alpha-2163), \frac{1}{97806723}(203984\alpha^2-5102488\alpha-20328135))$, $(\frac{1}{251001}(4697\alpha^2+8009\alpha-443037), \frac{1}{13972389}(-440621\alpha^2-66608\alpha+23555133))$
1333	$\chi_{31}^2\chi_{43}$	$(\frac{1}{27}(\alpha^2+11\alpha-3), \frac{1}{81}(10\alpha^2+218\alpha-3))$, $(\frac{1}{18}(-\alpha^2+3\alpha+360), \frac{1}{27}(8\alpha^2-2475))$
1359	$\chi_9\chi_{151}$	$(\frac{1}{142129}(296\alpha^2+5132\alpha-225320), \frac{1}{53582633}(23622\alpha^2-467154\alpha+14413916))$, $(\frac{1}{1847677}(3356\alpha^2-73516\alpha-2804192), \frac{1}{696574229}(-608886\alpha^2+270084\alpha+420037796))$
1477	$\chi_7^2\chi_{211}$	$(\frac{1}{4}(\alpha+16), \frac{1}{24}(-\alpha^2-14\alpha+96))$, $(\frac{1}{192}(-\alpha^2+4\alpha+384), \frac{1}{1536}(11\alpha^2-92\alpha-4800))$
2331	$\chi_9\chi_7\chi_{37}$	$(\frac{1}{470448}(773\alpha^2-8267\alpha-522852), \frac{1}{93148704}(-354307\alpha^2-3860291\alpha+137179020))$
2709	$\chi_9^2\chi_7\chi_{43}$	$(\frac{1}{1978032}(2285\alpha^2-126093\alpha-1325114), \frac{1}{401540496}(377025\alpha^2-20805345\alpha+492129022))$
2817	$\chi_9^2\chi_{313}$	$(\frac{1}{215168}(-1651\alpha^2+24709\alpha+736090), \frac{1}{211725312}(-105445\alpha^2+6447203\alpha+15509622))$

E24A6

Table 10: Results using Magma to perform descents on $E24A6$.

Conductor	Character	Generators
31	χ_{31}	$(\frac{1}{8}(3\alpha^2+17\alpha+68), \frac{1}{16}(-33\alpha^2-187\alpha-372))$, $(\frac{1}{32}(243\alpha^2+567\alpha+646), \frac{1}{128}(11097\alpha^2+32157\alpha+22842))$
67	χ_{67}	$(\frac{1}{225}(36\alpha^2-17\alpha+1181), \frac{1}{10125}(3025\alpha^2-48558\alpha-34552))$
133	$\chi_7\chi_{19}$	$(\frac{1}{2800}(-269\alpha^2-1635\alpha+35676), \frac{1}{28000}(-14257\alpha^2-86655\alpha+1228828))$

E37A1

Table 11: Results using Magma to perform descents on $E37A1$.

Conductor	Character	Generators
43	χ_{43}	$(0,0)$, $(\frac{1}{2}(-3\alpha^2-5\alpha+40), \frac{1}{2}(13\alpha^2+21\alpha-170))$, $(3\alpha^2-10\alpha+4, \frac{1}{2}(-61\alpha^2+213\alpha-108))$
61	χ_{61}	$(\frac{1}{12}(-5\alpha^2-4\alpha+96), \frac{1}{8}(9\alpha^2+6\alpha-185))$, $(\frac{1}{12}(-\alpha^2-2\alpha+27), \frac{1}{24}(-4\alpha^2+\alpha+66))$, $(0,0)$

Continued on next page

E37A1 Descent with Magma results – continued from previous page

Conductor	Character	Generators
103	χ_{103}	$(0,0)$, $(\frac{1}{4}(-31\alpha^2+27\alpha+1004), \frac{1}{8}(903\alpha^2-802\alpha-29194))$, $(\frac{1}{27239980563}(1356588440\alpha^2-3330300474\alpha-3775160249), \frac{1}{2595670507867707}(136700768167916\alpha^2-741678083944068\alpha-3156319688992064))$
127	χ_{127}	$(0,-1)$, $(\frac{1}{2}(\alpha^2-7\alpha+10), 4\alpha^2-27\alpha+40)$, $(\frac{1}{100}(-3\alpha^2-16\alpha+140), \frac{1}{1000}(-23\alpha^2-106\alpha+40))$
171	$\chi_9\chi_{19}$	$(\frac{1}{8975107815}(85515852\alpha^2-621376936\alpha+2124337399), \frac{1}{219540112262715}(-1735331969032\alpha^2+15407530457696\alpha-122636065963739))$, $(\frac{1}{8975107815}(72965876\alpha^2+573511012\alpha+2601236487), \frac{1}{43908022452543}(-408061382024\alpha^2-2551454733080\alpha-22209403640479))$, $(0,-1)$
247	$\chi_{13}^2\chi_{19}$	$(0,0)$, $(\frac{1}{32}(-\alpha^2+\alpha+82), \frac{1}{384}(17\alpha^2+15\alpha-1682))$, $(\frac{1}{3188646}(99221\alpha^2+470661\alpha-3632858), \frac{1}{2324522934}(76412311\alpha^2+618961863\alpha-2827016662))$
817	$\chi_{19}^2\chi_{43}$	$(0,-1)$, $(\frac{1}{228}(-14\alpha^2-101\alpha+2748), \frac{1}{152}(-14\alpha^2-101\alpha+2973))$, $(\frac{1}{1292769}(20068\alpha^2+218318\alpha-3854684), \frac{1}{1469878353}(-18483440\alpha^2-189696370\alpha+3278706427))$
853	χ_{853}	$(0,-1)$, $(\frac{1}{48}(-\alpha^2-5\alpha+309), \frac{1}{192}(11\alpha^2+55\alpha-3066))$, $(\frac{1}{144}(2\alpha^2+31\alpha-6), \frac{1}{576}(22\alpha^2+341\alpha-1641))$

E40A3

Table 12: Results using Magma to perform descents on $E40A6$.

Conductor	Character	Generators
7	χ_7	$(\frac{1}{344569}(30689\alpha^2-167080\alpha+119999), \frac{1}{202262003}(-118104208\alpha^2-59945097\alpha+33698073))$, $(-2\alpha^2-2\alpha+4, 4\alpha^2+2\alpha-9)$
63	$\chi_9\chi_7$	$(\frac{1}{18}(-\alpha^2+5\alpha+20), \frac{1}{18}(-\alpha^2+5\alpha+6))$, $(\frac{1}{316969}(22509\alpha^2-112014\alpha-21467), \frac{1}{178453547}(27371154\alpha^2-73569213\alpha-165786054))$
91	$\chi_7\chi_{13}$	$(\frac{1}{1058}(-163\alpha^2-277\alpha+2922), \frac{1}{24334}(-2007\alpha^2-1061\alpha+30156))$, $(\frac{1}{64}(7\alpha^2+54\alpha), \frac{1}{512}(-233\alpha^2-970\alpha+1728))$
223	χ_{223}	$(-\alpha^2+4\alpha+65, 6\alpha^2-17\alpha-344)$, $(\frac{1}{2}(\alpha^2-7\alpha-24), \frac{1}{2}(-13\alpha^2+63\alpha+586))$
259	$\chi_7\chi_{37}$	$(\frac{1}{108}(-\alpha^2+26\alpha+123), \frac{1}{648}(-10\alpha^2-133\alpha-63))$, $(\frac{1}{10368}(-163\alpha^2-101\alpha+17162), \frac{1}{746496}(9847\alpha^2-60559\alpha-1151522))$
499	χ_{499}	$(\frac{1}{12168}(-163\alpha^2-1391\alpha+33350), \frac{1}{949104}(13451\alpha^2+77935\alpha-2917342))$, $(\frac{1}{18252}(19\alpha^2+1524\alpha+21980), \frac{1}{1423656}(10781\alpha^2+153402\alpha+1029832))$
547	χ_{547}	$(\frac{1}{6561}(20\alpha^2-404\alpha+9153), \frac{1}{177147}(2878\alpha^2-28684\alpha-193590))$, $(\frac{1}{36}(\alpha^2-8\alpha+52), \frac{1}{216}(23\alpha^2-184\alpha-361))$