2016-02-12

William A. Stein

2/12/2016

Contents

1 February 12, 2016: Elliptic curves (part 1)

1.1 William Stein

(reminder: Go to Brian Conrad's talk on ABC in MEB right after class.)

Elliptic curves (and modular forms) are absolutely central in number theory:

Fermat's Last Theorem, congruent number problem, Birch and Swinnerton-Dyer conjecture, CM elliptic curves (class field theory), one-dimensional abelian varieties.

Unsolved problem: Is there an algorithm that decides whether or not a cubic $F(x, y) \in [x, y]$ has a rational solution?

(Answer: conjecturally yes, but we don't know!)

Unsolved problem: Is there an algorithm to determine whether or not an integer n is the area of a right triangle with rational side lengths?

(Answer: conjecturally yes.)

Sage has a ridiculous amount of functionality for computing with elliptic curves.

There used to be several programs relevant to computing with elliptic curves (e.g., SIMATH), but now the ones that matter are Sage and Magma:

1.1.1 First rate support for elliptic curves:

- Sage (and PARI/mwrank, which are both in Sage)
- Magma

Also relevant: smalljac for point counting

1.1.2 Third rate (or worse) for elliptic curves:

- Maple: has a package called apecs from the old days. Find the pdf file with google. I don't know if anybody uses this anymore.
- Mathematica I've never heard of anybody doing anything useful with elliptic curves in Mathematica (and couldn't find anything via Google searches).
- Matlab/Mupad ?

1.2 2. Quick background on elliptic curves

An elliptic curve E over a field k is a genus one curve with a distinguished rational point. Such a thing can be given by an equation

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

In Sage (or Magma), make an elliptic curve by typing EllipticCurve([1,2,3,4,6]):

```
EllipticCurve([1,2,3,4,6])
```
```
Elliptic Curve defined by y^2 + x*y + 3*y = x^3 + 2*x^2 + 4*x + 6 over Rational Field
```

```
%magma
EllipticCurve([1,2,3,4,6])
```
```
Elliptic Curve defined by y^2 + x*y + 3*y = x^3 + 2*x^2 + 4*x + 6 over Rational Field
```

```
%gp
ellinit([1,2,3,4,6])
```
```
[1, 2, 3, 4, 6, 9, 11, 33, 44, -183, -4293, -14212, 6128487/14212, Vecsmall([1]),
[Vecsmall([128, -1])], [0, 0, 0, 0, 0, 0, 0, 0]]
```

You can also make elliptic curves in a few other ways:

```
EllipticCurve_from_    # [put cursor after _ and press tab]
```

```
E = EllipticCurve_from_j(2016)
E
```
```
Elliptic Curve defined by y^2 = x^3 - 21*x - 14 over Rational Field
```

The j-invariant of an elliptic curve is an invariant of the isomorphism class of the curve over the algebraic closure. It's just an algebraic function of the coefficients. Here it is in general below:

```
E.j_invariant()
```
```
2016
```

```
R.<a1,a2,a3,a4,a6> = QQ[]
E_generic = EllipticCurve(R.gens())
E_generic
```
```
Elliptic Curve defined by y^2 + a1*x*y + a3*y = x^3 + a2*x^2 + a4*x + a6 over Multivariate
Polynomial Ring in a1, a2, a3, a4, a6 over Rational Field
```

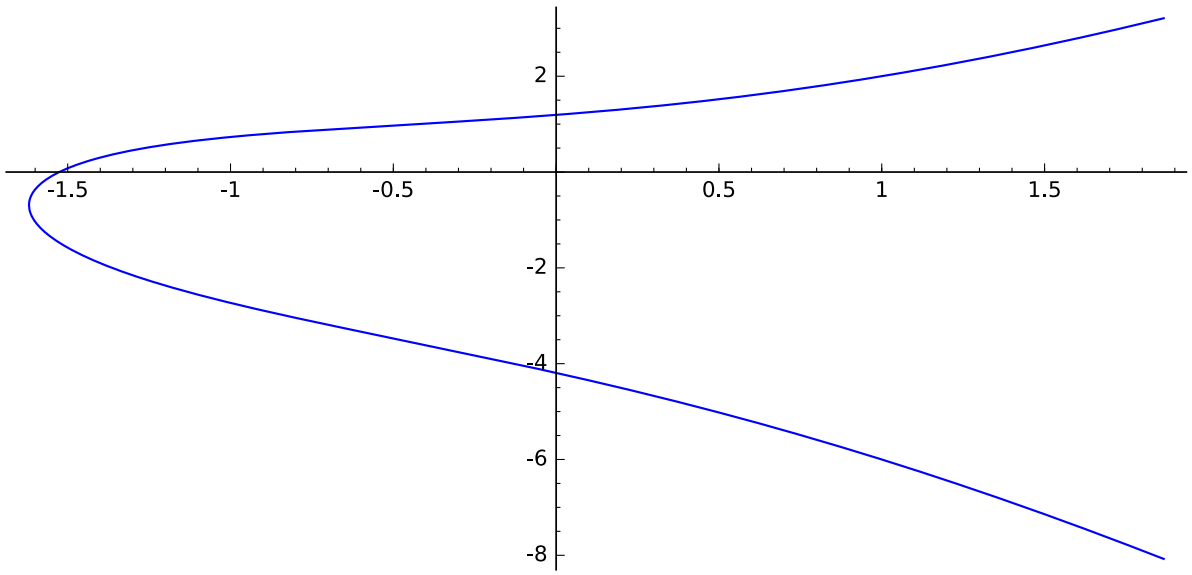
```
show(E_generic.j_invariant())
```
$$\frac{a_1^{12} + 24a_1^{10}a_2 + 240a_1^8a_2^2 - 72a_1^9a_3 + 1280a_1^6a_2^3 - 1152a_1^7a_2a_3 - 144a_1^8a_4 + 3840a_1^4a_2^4 - 6912a_1^5a_2^2a_3 + 1728}{-a_1^4a_2a_3^2 + a_1^5a_3a_4 - a_1^6a_6 - 8a_1^2a_2^2a_3^2 + a_1^3}$$

There's elliptic curves over any field

```
EllipticCurve(GF(9,'a'), [1,2,3,4,5])
```
```
Elliptic Curve defined by y^2 + x*y = x^3 + 2*x^2 + x + 2 over Finite Field in a of size
3^2
```

```
EllipticCurve(RR, [1,2,3,4,5])
```
```
Elliptic Curve defined by y^2 + 1.00000000000000*x*y + 3.00000000000000*y = x^3 +
2.00000000000000*x^2 + 4.00000000000000*x + 5.00000000000000 over Real Field with 53 bits
of precision
```

```
# They are NOT ellipses; instead, they come up when computing arc \
   lengths of ellipses...
EllipticCurve(RR, [1,2,3,4,5]).plot()
```



There are a million things associated to an elliptic curve. Galois representations, groups, number fields, towers of GL2 extensions, and the list goes on for a very long time

```
E = EllipticCurve_from_j(2016)
E.conductor()
```
```
508032
```
```
E.rank()
```
```
0
```

```
L = E.lseries()
L(1)
```
```
2.34703797000777
```

```
E.analytic_rank()
```
```
0
```
```
E.torsion_order()
```
```
1
```

```
E.division_field(2, 'a')
```
```
Number Field in a with defining polynomial x^6 - 48*x^4 - 28*x^3 + 453*x^2 + 420*x - 526
```
```
%time E.division_field(3, 'a')
```
```
Number Field in a with defining polynomial x^48 - 24*x^47 - 876*x^46 + 15736*x^45 +
389094*x^44 - 1128480*x^43 - 118733264*x^42 - 1541713176*x^41 + 20161182645*x^40 +
413890125680*x^39 + 122478216504*x^38 + 2331515874936*x^37 - 321414075366214*x^36 -
12108052719896136*x^35 - 106208488655746716*x^34 + 193391383126374464*x^33 +
3596283559135581971 4*x^32 + 467360144091890455896*x^31 - 116566744482362091726 8*x^30 -
695997631604869108697 04*x^29 - 443431200882789341857650*x^28 +
243221625092532551693771 2*x^27 + 510567605519274862760014 80*x^26 -
516030896919922473854637 6*x^25 - 482813654065281818918218 2083*x^24 +
894711474456520205883107 9088*x^23 + 707463010704999071433206 720088*x^22 +
314374232682172784857593 981224*x^21 - 369298230995022550506747 26945514*x^20 -
342073127180645251965682662618936*x^19 + 127196265982491030879611 1820912188*x^18 +
333738046109422165677942721088289 60*x^17 + 219141392390956066658265143086409 46*x^16 -
238642463246896485555776566709718 7800*x^15 + 243577441743827205247744963499338 8852*x^14 +
250173900113857713806643262759969215144*x^13 +
861154242273006785258545692895697667810*x^12 -
108118531642355360426939907069762611375 52*x^11 -
499491432140379615683632180026142316725 68*x^10 +
202092629996850183287450573302939532358600*x^9 +
274155656990718398419408168102883169912938 1*x^8 +
206044100031167368085801584850407031850336 0*x^7 -
```

```
172915186786513676432039348472258565542213 12*x^6 -
3553495275194505038829870859969546758831786 48*x^5 +
5409927268774861859592812881127959571978411 34*x^4 +
6470928968998811128619821455480530272176191 544*x^3 +
8938616413372307641583398985996103047669892 132*x^2 +
7630043609842305434669235334544158234894321 7232*x +
4606923799234915412184288395989932535649896 89609

CPU time: 1.80 s, Wall time: 1.94 s
```

```
list(E.isogeny_class())
[Elliptic Curve defined by y^2 = x^3 - 21*x - 14 over Rational Field]
```

Birch and Swinnerton-Dyer Conjecture: For elliptic curves over we have E.analytic_rank() == E.rank()

Exercise: Make up a random curve right now and try to use Sage to verify the BSD rank conjecture for it (assuming that the output of Sage is correct).

1.3 3. Tables of Elliptic Curves

1.3.1 3.1 Antwerp

There is a long history of people making systematic tables of elliptic curves, starting in Belgium in the 1970s.

- Antwerp: http://wstein.org/Tables/antwerp/index.html made using this computer.

Look at http://wstein.org/Tables/antwerp/reliability/ about making these tables.

1.3.2 3.2 Cremona's tables

Cremona has spent 25 years expanding the Antwerp tables approach. He now has every single elliptic curve up to conductor 370000, with extensive data about each. His tables are in Sage (they get regularly updated).

```
C = CremonaDatabase(); C
Cremona's database of elliptic curves with conductor at most 359999
```

```
C.number_of_curves()
2247187
```

```
C.number_of_isogeny_classes()
```

```
1569126
```

```
i = 0
for E in C:
    print E.conductor(), E.cremona_label(), E
    i += 1
    if i > 10:
        break
```

```
11
 11a1 Elliptic Curve defined by y^2 + y = x^3 - x^2 - 10*x - 20 over Rational Field
11 11a2 Elliptic Curve defined by y^2 + y = x^3 - x^2 - 7820*x - 263580 over Rational
Field
11 11a3 Elliptic Curve defined by y^2 + y = x^3 - x^2 over Rational Field
14 14a1 Elliptic Curve defined by y^2 + x*y + y = x^3 + 4*x - 6 over Rational Field
14 14a2 Elliptic Curve defined by y^2 + x*y + y = x^3 - 36*x - 70 over Rational Field
14 14a3 Elliptic Curve defined by y^2 + x*y + y = x^3 - 171*x - 874 over Rational Field
14 14a4 Elliptic Curve defined by y^2 + x*y + y = x^3 - x over Rational Field
14 14a5 Elliptic Curve defined by y^2 + x*y + y = x^3 - 2731*x - 55146 over Rational Field
14 14a6 Elliptic Curve defined by y^2 + x*y + y = x^3 - 11*x + 12 over Rational Field
15 15a1 Elliptic Curve defined by y^2 + x*y + y = x^3 + x^2 - 10*x - 10 over Rational
Field
15 15a2 Elliptic Curve defined by y^2 + x*y + y = x^3 + x^2 - 135*x - 660 over Rational
Field
```

Exercise: Use Cremona's tables to determine how many distinct isomorphism classes of elliptic curves have conductor 2016. Will there be any curves for next year?

```
factor(2016)
```

```
2^5 * 3^2 * 7
```

```
# hint
C[11]
```

```
{'allcurves': {'a1': [[0, -1, 1, -10, -20], 0, 5], 'a3': [[0, -1, 1, 0, 0], 0, 5], 'a2':
[[0, -1, 1, -7820, -263580], 0, 1]}, 'allbsd': {'a1': [5, 1.26920930427955,
0.253841860855911, 1.0, 1], 'a3': [1, 6.34604652139777, 0.253841860855911, 1.0, 1], 'a2':
[1, 0.253841860855911, 0.253841860855911, 1.0, 1]}, 'degphi': {'a1': 1}, 'allgens': {'a1':
[], 'a3': [], 'a2': []}}
```

1.3.3 3.3 Stein-Watkins tables

In 2001 at Harvard I started making large tables of elliptic curves by just writing down lots of curves. Mark Watkins got involved and wrote lots of very fast code. Together we made the SteinWatkins tables, which are a large table of curves of conductor up to 10^9. We don't have all curves up to that bound just a lot. We also have curves of prime conductor up to some huge bound.

```
i = 0
```

```
for E in SteinWatkinsAllData(10):
    print E.curves
    print EllipticCurve(E.curves[0][0]).conductor()
    i += 1
    if i > 5:
        break
```

```
[[[1, 1, 0, -63, -1539], '(3,6,1)', 'X', '1']]
1000002
[[[1, 1, 1, -1843618, 962738063], '[5,1,1]', '1', '1']]
1000002
[[[1, 1, 1, -567, -147], '[5,7,1]', '1', '1']]
1000002
[[[1, 0, 1, -364, -1366], '[9,3,1]', 'X', '1']]
1000002
[[[0, 1, 0, 10300, -1000108], '(8,6,1)', 'X', '1']]
1000004
[[[0, 1, 1, -831, -9475], '[3,3,1,1]', '1', '1']]
1000005
```

```
i = 0
for E in SteinWatkinsPrimeData(1):
    print E.curves
    print EllipticCurve(E.curves[0][0]).conductor()
    i += 1
    if i > 5:
        break
```

```
[[[1, 0, 1, -472, -3951], '[1]', '1', '1']]
100000937
[[[0, -1, 1, 97, 280], '(1)', 'X', '1']]
100000963
[[[1, 0, 0, -75, 536], '(1)', 'X', '1']]
100001183
[[[1, 0, 0, -120, -167], '[1]', 'X', '1']]
100001399
[[[0, -1, 1, -22, 490], '(1)', 'X', '1']]
100002251
[[[0, 1, 1, -180, 738], '[1]', 'X', '1']]
100002629
```

1.3.4 3.4 New table: all 238,764,310 curves of naive height up to $2.9 \cdot 10^{10}$

http://wstein.org/papers/2016-height/

Not in Sage. Is maybe 30GB or so

1.4 4. Magma versus Sage

If you want to do a range of explicit computations with elliptic curves, you will very likely use Sage or Magma. If you're really serious, you'll use both.

Note, as mentioned above, that both Sage and Magma are far ahead of all other software for elliptic curves.

- Magma reference manual about elliptic curves.
- Sage reference manual about elliptic curves.
- Pari reference manual about elliptic curves. pari is part of Sage and has some unique powerful functionality, e.g., ellheegner

1.4.1 History:

When I started contributing to Magma in 1999, I remember that it was way, way behind PARI. I remember having lunch with John Cannon (founder of Magma), and telling him we could "kill Pari" if only Magma would have dramatically faster code for computing $a_p = p + 1 - \#E(_p)$; he responded "I don't want to kill Pari."

In any case, a few years later, John wisely hired Mark Watkins to work fulltime on Magma, and Mark has been working there for over a decade. Mark is definitely one of the top people in the world at implementing (and using) computational number theory algorithms, and he's ensured that Magma can do a lot. Some of that "do a lot" means catching up with (and surpassing!) what was in Pari and Sage for a long time (e.g., point counting, p-adic L-functions, etc.)

However, in addition, many people have visited Sydney and added functionality for doing higher descents to Magma, which is not available in any open source software. Search for Magma in this paper to see how, even today, there seems to be no open source way to compute the rank of the curve $y^2 = x^3 + 169304x + 25788938$. Yes, this makes me pissed off.

1.4.2 Situation today

There are several elliptic curves algorithms available only in Magma (e.g., higher descents) and some available only in Sage (L-function rank bounds, some overconvergent modular symbols, zeros of L-functions, images of Galois representations). I could be wrong about stuff not in Sage, since WITH MONEY almost anything can get implemented in a year And Magma has money.

The code bases are completely separate, which is a very good thing. Any time something gets implemented in one, it gets (or should get) tested via a big run through tables of elliptic curves up to some bound. This usually results in numerous bugs being found. I remember refereeing the "integral points" code in Sage by running it against all curves up to some bound and comparing to what Magma output, and getting many discrepancies, which showed that there were bugs in both Sage and Magma.

```
# My fav curve:
E = EllipticCurve('389a')
show(E)
```

$y^2 + y = x^3 + x^2 - 2x$

```
E.ainvs()
```

```
(0, 1, 1, -2, 0)
```

```
E.rank()
```
```
2
```

```
E.integral_points()
```
```
[(-2 : 0 : 1), (-1 : 1 : 1), (0 : 0 : 1), (1 : 0 : 1), (3 : 5 : 1), (4 : 8 : 1), (6 : 15 :
1), (39 : 246 : 1), (133 : 1539 : 1), (188 : 2584 : 1)]
```

```
%magma
E := EllipticCurve([0,1,1,-2,0]);
print Rank(E);
print IntegralPoints(E);
```
```
2
[ (-2 : 0 : 1), (-1 : -2 : 1), (0 : -1 : 1), (1 : -1 : 1), (3 : -6 : 1), (4 : 8 : 1), (6 :
15 : 1), (39 : -247 : 1), (133 : -1540 : 1), (188 : -2585 : 1) ]
[ <(-2 : 0 : 1), 1>, <(-1 : -2 : 1), 1>, <(0 : -1 : 1), 1>, <(1 : -1 : 1), 1>, <(3 : -6 :
1), 1>, <(4 : 8 : 1), 1>, <(6 : 15 : 1), 1>, <(39 : -247 : 1), 1>, <(133 : -1540 : 1), 1>,
<(188 : -2585 : 1), 1> ]
```

Exercise: 1. Make up a few elliptic curves and compute the integral points on them.

1. Try to find an elliptic curve with at least 12 integral points on it. You can search using Google if you want or whatever.