5.

楕円曲線の計算法入門：実践編

横山 俊一[*1]（九州大学）

本稿は，第 25 回整数論サマースクール「楕円曲線とモジュラー形式の計算」における筆者の火曜午前の同題目の講演に基づいている．ここでは楕円曲線に纏わる基本的な計算法について，基礎体が 1) 有理数体 $\mathbb{Q}$ と 2) 代数体 $K/\mathbb{Q}$ の場合を解説する（一部有限体 $\mathbb{F}_p$ の場合も扱う）．また講演では紹介できなかった 3) 超楕円曲線の場合についても解説する．実際の講演で用いたデモプログラムは筆者の web ページから入手可能である[*2]．本稿では（一部修正しているが）主にこのデモプログラムに従って解説を進める．

以下，全文を通して CAS（計算代数システム）は **Magma** を用いる．Magma は豪シドニー大学を中心ととして開発・運営されている計算代数システムである．正式リリースは 1993 年で，その前身は計算代数システム Cayley (1982-1993) である．Magma という名前は，ブルバキ流の亜群 *groupoid* の別名称が由来である（つまり頭文字をとった何らかの略称ではない）．圏論 *category theory* に基づいた代数構造から従う内部関数の定義方法により，直感的にプログラミングを行うことが可能となっている．

Magma は有償ソフトウェアである[*3]（目安として 1 ライセンス 10 万円強で，少なくとも 3 年間はアップデート権を保持できる）．しかし **Magma Calculator** とよばれるオンライン無料体験版が公開されている．

`http://magma.maths.usyd.edu.au/calc/`

[*1] 本稿に関連する研究は JSPS 科研費 JP15K17515 の助成を受けたものです．
This work is supported by JSPS KAKENHI Grant Number JP15K17515.

[*2] `http://www2.math.kyushu-u.ac.jp/~s-yokoyama/files/ss2017demo.txt`

[*3] 有償ではあるが，プロプライエタリライセンスとして提供されている．“Cost Recovery” を目的としており，営利目的ではなく開発・運営の費用を回収するための価格設定である．

Magma は常に最新版に更新されており（2018/01/01 現在 ver.2.23），50KB までの入力に対して 120 秒（通信時間等を含む）までの計算が（複数回[*4]）可能である．

リファレンスマニュアルは web ページから無料で入手可能（英語版のみ）であり，カテゴリー別検索・辞書式検索のどちらも

`http://magma.maths.usyd.edu.au/magma/documentation/`

から利用できる．なお HTML ベースで閲覧することを推奨する（実際に Magma を購入すると pdf 版も利用できるが，数千ページの pdf から逆引きするのはむしろ不便である）．また，最初にチュートリアル “First Steps in Magma” に目を通しておくとよい．加えて，書籍 “Discovering Mathematics with Magma”（Springer, 2006）の appendix にもチュートリアルが掲載されている．

5.1 Magma の文法・簡単な計算

まず Magma の文法・扱い方に慣れよう．本稿では Magma における入力と出力を以下のように表記する．

```
> 1+2+3;
6
```

1 行目が入力，2 行目が出力である．なお 1 行目最初の > は実際に入力する必要はない．

```
> Factorization(20170829);
[ <7, 1>, <2881547, 1> ]
```

関数 Factorization の引数が自然数の場合，その素因数分解を出力する．表示はタプル *tuple* で与えられる．例えば [<2, 3>, <17, 1>] ならば $2^3 \cdot 17$ のことである．

```
> SetVerbose("Factorization",1);
> n:=155091189582461323967422273374060483191006034694003396 14
>    469771087037330452965523211078730059982890210854424804 93
>    813775112197024286939278796430189587028111515759385199 84
>    102178816;
```

[*4] 明示的には回数制限はないが，同一 IP から大量にジョブが投げられた場合は通信を制限される可能性がある．

```
> Factorization(n);
Integer main factorization (primality of factors will be proved)
Effort: 3
Seed: 1886283651 0
    Number: 155...（中略）...816
Pollard Rho
    Trials: 8191
    Number: 231...（中略）...251 (105 digits)
    Factor: 21727894856911 (14 digits)
    Cofactor: 106...（中略）...941 (92 digits)
    Time: 0.010
Pollard Rho
    Trials: 8191
    Number: 21727894856911 (14 digits)
    Factor: 3726911 (7 digits)
    Cofactor: 5830001 (7 digits)
    Time: 0.000
Pollard Rho
    Trials: 8191
    Number: 3726911 (7 digits)
Pollard Rho
    Trials: 8191
    Number: 5830001 (7 digits)
Pollard Rho
    Trials: 8191
    Number: 106...（中略）...941 (92 digits)
    No factor found
    Time: 0.000

1 composite number remaining

ECM
    x: 106...（中略）...941 (92 digits)
    Initial B1: 5000, limit: 858248
    Initial Pollard p - 1, B1: 45000
    Step 1; B1: 5000 [858248], digits: 92, elapsed time: 0.029
    Factor: 146234082633259 (15 digits)
```

```
    Cofactor: 729...（中略）...799 (77 digits)
    Total ECM time: 0.220
ECM
    x: 729...（中略）...799 (77 digits)
    Initial B1: 5429, limit: 176526
    Step 1; B1: 5429 [176526], digits: 77, elapsed time: 0.000
    Step 10; B1: 6106 [176526], digits: 77, elapsed time: 0.240
    Step 20; B1: 6906 [176526], digits: 77, elapsed time: 0.550
    Step 30; B1: 7756 [176526], digits: 77, elapsed time: 0.889
    Factor: 75045259055838983 (17 digits)
    Cofactor: 971...（中略）...953 (60 digits)
    Total ECM time: 0.919
ECM
    x: 971...（中略）...953 (60 digits)
    Initial B1: 7756, limit: 16224
    Step 1; B1: 7756 [16224], digits: 60, elapsed time: 0.000
    Factor: 6722279202985811 (16 digits)
    Cofactor: 144526707646708212356380247022426585248301323
    (45 digits)
    Total ECM time: 0.040

Total time: 1.199

[ <2, 206>, <67, 2>, <271, 1>, <5351, 1>, <3726911, 1>,
  <5830001, 1>, <146234082633259, 1>, <6722279202985811, 1>,
  <75045259055838983, 1>,
  <144526707646708212356380247022426585248301323, 1> ]
```

上は 177 桁の自然数 n の素因数分解である．関数 `SetVerbose` は，内部で行われている計算の詳細を出力させるためのものである．ここでは関数 `Factorization` の出力レベルを 1 にせよという指示を出している（出力レベルのデフォルト値は 0 で，この場合は結果のみを出力する）．Magma では

- Pollard ρ 法
- 楕円曲線法（ECM）
- 多変数多項式二次篩（MPQS）法

の 3 種類のアルゴリズムが採用されており，原則として上から順に試行しなが

ら分解を行う．つまり今回の場合は，まず Pollard ρ 法で分解を試み，それでも残った巨大な合成数（と思われる数[*5]）を楕円曲線法で分解していることが分かる．出力レベルをデフォルト値に戻してから，計算を続けることにする．

```
> SetVerbose("Factorization",0);
```

関数 Factorization は，多項式の因数分解にも利用できるが，一つだけ注意点がある．

```
> Factorization(x^3+6*x^2+11*x+6);


>> Factorization(x^3+6*x^2+11*x+6);
                  ^
User error: Identifier 'x' has not been declared or assigned
```

一変数多項式 $x^3+6x^2+11x+6$ を因数分解させようとすると，上記のようなエラーメッセージが表示される．これは「変数 x が未定義（何者か指示されていない）ゆえ，計算できない」というエラーである．Magma では通常の（$=\mathbb{Q}$ 上の）因数分解だけではなく，拡大体 $K/\mathbb{Q}$ 上や有限体 $\mathbb{F}_p$ 上での因数分解もサポートしているため，利用者が「どの体上で因数分解しているのか」を誤らないよう，このように strict な取り決めが存在する[*6]．この場合は

```
> P<x>:=PolynomialRing(Integers());
```

として，x が整数係数の一変数多項式環の生成元であると宣言すればよい．

```
> Factorization(x^3+6*x^2+11*x+6);
[
    <x + 1, 1>,
    <x + 2, 1>,
    <x + 3, 1>
]
```

関数 Factorization に関するエラーをもう一例紹介する．以下は自然数の商 $10000000/20=500000$ の素因数分解である．

```
> Factorization(10000000/20);


>> Factorization(10000000/20);
```

[*5] この時点では合成数かどうか不明である．実際は素数かもしれない．

[*6] これは最初やや面倒に感じるが，使い込み始めるとその有難みが実感できる．

```
                ^
Runtime error in 'Factorization': Bad argument types
Argument types given: FldRatElt
```

“Bad argument types” エラーは「本来入力されるべき値や多項式が入っていない」という意味であり，ここでは「入力が自然数ではない」ことが原因である．その直後に “Argument types given: FldRatElt” と出力されており，つまり「10000000/20 は 有理数 であるから分解できない」と言っているわけである．FldRatElt は Field of Rational, Element の略である．実際にはこの値は 500000 であるから自然数と認識すべきであるが，Magma では（というより Magma 以外の計算代数システムにおいても）除算を行った時点で有理数の元と解釈されてしまうのである．これを回避するには，除算の結果が再び自然数（ここでは整数で十分）であると宣言してやればよい．

```
> Factorization(Integers()!(10000000/20));
[ <2, 5>, <5, 6> ]
```

`X!n` が「`n` を `X` の元であるとみなす」という意味である．このテクニックは，lifting や reduction を多用する数論の計算において非常に重宝する．

```
> Parent(10000000.0/20.0);
Real field of precision 30
```

なお小数点を付けると，有理数ではなく実数とみなされる．計算機において実数を扱う場合は精度 *precision* をつける必要があるので，計算誤差に注意する必要がある．デフォルト値は 30 であるが，好きな値に変更できる．

```
> K<a>:= QuadraticField(5); a; Parent(a);
a
Quadratic Field with defining polynomial x^2 - 5 over the
Rational Field
> a^2-5; Sqrt(5)^2-5;
0
0.00000000000000000000000000000
> a^2-5 eq Sqrt(5)^2-5;


>> a^2-5 eq Sqrt(5)^2-5;
          ^
Runtime error in 'eq': Bad argument types
Argument types given: FldQuadElt[FldRat], FldReElt
```

二次体 $K = \mathbb{Q}(\sqrt{5})$ を与えて，その生成元を a とする．つまり a $= \sqrt{5}$ である．Magma には平方根を求める関数 Sqrt があるが，これが実数値近似を返す関数であるのに対して a は exact に $\sqrt{5}$ を保持していることに注目である．即ち a を 2 乗して 5 を引くと exact に 0 となるが，Sqrt(5) を 2 乗して 5 を引くと 0.000... となり，厳密にはこれは 0 ではない（打ち切り誤差の可能性を捨てきれない）．従って，両者がイコールであるか？という問いはナンセンスであり，上記のようなエラーメッセージが出力される．

```
> K:=GF(7); s:=K!23; s; Parent(s);
2
Finite field of size 7
```

有限体 $\mathbb{F}_p$（より一般に $\mathbb{F}_q$；q は素数べき）も扱える．上は $\mathbb{F}_7$ において 23 が 2 であることを示している．なお関数 FiniteField と GF は同じ関数である．

```
> &+[k^2 : k in [1..24]];
4900
```

上は $\sum_{k=1}^{24} k^2$ を計算している．&+ を &* とすれば $\prod_{k=1}^{24} k^2$ を計算できる．

```
> exists(x,y){<x,y> : x,y in [1..10] | x^2+y^2 eq 89};
true
> x; y;
5
8
```

これは $1 \leq x, y \leq 10$ において $x^2 + y^2 = 89$ をみたすような $(x, y) \in \mathbb{Z}^{\oplus 2}$ が存在するかを調べる関数である．もし一つでも見つかれば，その瞬間に計算を停止して x, y に値を格納する．ここでは $x = 5$, $y = 8$ が代入されている．もし一つも見つからなければ false を返す．exists の代わりに forall とすれば，$1 \leq x, y \leq 10$ なる全ての $(x, y) \in \mathbb{Z}^{\oplus 2}$ に対して $x^2 + y^2 = 89$ が成り立つかを判定する真偽判定（T/F）関数となる（この場合はもちろん false を返す）．

```
> C<i> := QuadraticField(-1);
> conj := hom<C -> C | -i>;
> conj(3-4*i);
4*i + 3
```

Magma は圏論に基づいた設計であるため，写像の概念も自然に扱える．例え

ば $C = \mathbb{Q}(\sqrt{-1})$ として，C 上の準同型写像を conj と定義する．準同型性を仮定しない，単なる写像であれば hom の代わりに map とする．その後の -i の部分は，C の生成元 i（ここでは虚数単位）を $-i$ に写すという意味である．つまりこれは複素共役を与える写像である．試しに $3-4i$ を入力すれば，降べき・昇べきが入れ替わるが，確かに複素共役 $3+4i$ が出力される．

```
> G<a,b>:=Sym(4); // symmetric group on 4 elements
> Order(a) eq 4 or Order(b) eq 4;
true
> IsAlternating(G);
false
```

Magma では群論の計算も容易に実行できる（後に楕円曲線の Mordell-Weil 群や単数群などを扱うので，この機能は必須である）．上では 4 次対称群 S_4 を定義し，その 2 つの生成元のどちらかの位数が 4 であることを確かめている．また S_4 が交代群ではないことを確認している．Is... の形の関数は他にも IsSymmetric, IsAbelian, IsNormal, IsSoluble などがある．

```
> P<x>:=PolynomialRing(Rationals());
> f:=x^6+x^4-2*x^2-1;
> Gf:=GaloisGroup(f); Gf;
Permutation group Gf acting on a set of cardinality 6
Order = 12 = 2^2 * 3
    (2, 5)(3, 6)
    (1, 4)(3, 6)
    (1, 5, 3)(2, 6, 4)
> IsAbelian(Gf);
false
```

これは Galois 群の計算である．$\mathbb{Q}$ 上 6 次拡大体なので Galois 群は S_6 の部分群として得られ，その位数は 12 である．また Galois 群はアーベル群ではないことも分かる．

```
> IsIsomorphic(Gf,AlternatingGroup(4));
true Mapping from: GrpPerm: Gf to GrpPerm: $, Degree 4,
Order 2^2 * 3
Composition of Mapping from: GrpPerm: Gf to GrpPC and
Mapping from: GrpPC to GrpPC and
Mapping from: GrpPC to GrpPerm: $, Degree 4, Order 2^2 * 3
```

実はこの Galois 群は 4 次交代群 A_4 と同型である．Magma では，それを確かめるための関数 `IsIsomorphic` を実行すると，返り値 `true` と同時に Galois 群から A_4 への同型写像を与える．ここで最小分解体を求め，その上で定義多項式 `f` を因数分解してみると，以下のように一次式の積に分解される．

```
> S<b>:=SplittingField(f); S;
Number Field with defining polynomial x^12 + 4*x^10 + 10*x^8
+ 34*x^6 - 7*x^4 + 98*x^2 + 49 over the Rational Field
> Factorization(PolynomialRing(S)!f);
[
    <$.1 + 1/22806*(-191*b^10 - 295*b^8 - 111*b^6 - 908*b^4
     + 19089*b^2 - 16576), 1>,
    <$.1 + 1/22806*(191*b^10 + 295*b^8 + 111*b^6 + 908*b^4
     - 19089*b^2 + 16576), 1>,
    <$.1 + 1/22806*(-284*b^11 - 1138*b^9 - 3261*b^7
     - 11090*b^5 + 2277*b^3 - 42259*b), 1>,
    <$.1 + 1/22806*(-284*b^11 - 1138*b^9 - 3261*b^7
     - 11090*b^5 + 2277*b^3 - 19453*b), 1>,
    <$.1 + 1/22806*(284*b^11 + 1138*b^9 + 3261*b^7
     + 11090*b^5 - 2277*b^3 + 19453*b), 1>,
    <$.1 + 1/22806*(284*b^11 + 1138*b^9 + 3261*b^7
     + 11090*b^5 - 2277*b^3 + 42259*b), 1>
]
```

5.2 $\mathbb{Q}$ 上の楕円曲線

それでは楕円曲線の計算に移ろう．まずは $\mathbb{Q}$ 上の楕円曲線

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

を考える．5 つの係数を順に入力して，楕円曲線 $E/\mathbb{Q}$ を得る．

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

とくに簡略化された形

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

の楕円曲線を扱う場合は `EllipticCurve([a,b]);` とすればよい．

```
> bInvariants(E);
[ 9, 11, 33, 44 ]
> cInvariants(E);
[ -183, -4293 ]
> jInvariant(E);
6128487/14212
> Discriminant(E);
-14212
> Conductor(E);
7106
> Factorization($1);
[ <2, 1>, <11, 1>, <17, 1>, <19, 1> ]
```

各種不変量なども計算できる．最後の入力に現れる `$1` は 1 つ前の出力を表し，ここでは導手の値 7106 とみなされている[*7]．導手の素因数分解を見ると，この楕円曲線の bad prime は $2, 11, 17, 19$ の 4 つであることが分かる．これらは関数 `BadPrimes` でも求められる．

```
> BadPrimes(E);
[ 2, 11, 17, 19 ]
```

Magma には，Cremona による楕円曲線のデータベースが含まれている．そのため，例えば導手を指定して楕円曲線のリストを得ることも可能である．

```
> DB:=EllipticCurveDatabase(); DB;
John Cremona's elliptic curve database
```

例えばこのデータベースから，導手 37 の楕円曲線を引いてみよう．

```
> ES:=EllipticCurves(DB,37);
> ES;
[
    Elliptic Curve defined by y^2 + y = x^3 - x over
    Rational Field,
    Elliptic Curve defined by y^2 + y = x^3 + x^2 - 23*x - 50
    over Rational Field,
    Elliptic Curve defined by y^2 + y = x^3 + x^2 - 1873*x
```

[*7] 同様にして `$2` とすれば 2 つ前の出力，即ち判別式 -14212 が引用される．なお前節の最小分解体上での定義多項式の因数分解で出力された `$.1` は単なる未定義の変数を表すため，この記号 `$1` とは全く異なるので注意である．

```
    - 31833 over Rational Field,
    Elliptic Curve defined by y^2 + y = x^3 + x^2 - 3*x + 1
    over Rational Field
]
> Conductor(ES[1]);
37
```

Cremona's index（楕円曲線のラベル記号）を使うこともできる．例えば "37a1" と書いたら，導手 37 の楕円曲線のうち，同種類別して得られた a1 という曲線を指す．

```
> E37:=EllipticCurve("37a1");
> E37;
Elliptic Curve defined by y^2 + y = x^3 - x over
Rational Field
```

続いて Mordell-Weil 群（楕円曲線の有理点全体のなす有限生成アーベル群）$E(\mathbb{Q})$ を求める．先ほどの楕円曲線 E を引き続き用いる．

```
> time MordellWeilGroup(E);
Abelian Group isomorphic to Z
Defined on 1 generator (free)
Mapping from: Abelian Group isomorphic to Z
Defined on 1 generator (free) to Set of points of E with
coordinates in Rational Field given by a rule [no inverse]
true true
Time: 0.290
> TorsionSubgroup(E);
Abelian Group of order 1
```

この場合は $E(\mathbb{Q}) \simeq \mathbb{Z}$ であることが分かる．さらにこの楕円曲線に対する Tate-Shafarevich 群は自明となるから，2-Selmer 群は $\mathbb{Z}/2\mathbb{Z}$ となる．

```
> TwoSelmerGroup(E);
Abelian Group isomorphic to Z/2
Defined on 1 generator
Relations:
    2*$.1 = 0
Mapping from: Univariate Quotient Polynomial Algebra in theta
over Rational Field
```

```
with modulus theta^3 - 3*theta^2 + 64*theta + 256 to Abelian
Group isomorphic to Z/2
Defined on 1 generator
Relations:
    2*$.1 = 0 given by a rule
```

具体的な生成元は射影座標で与えられる．ここでは $E(\mathbb{Q})$ の生成元として $(-1,-3)$ が選択され，その位数は無限大であることが確かめられる．

```
> pt:=Generators(E)[1]; pt;
(-1 : -3 : 1)
> Order(pt1);
0
```

位数が 0 となっているが，これは無限大を意味していることに注意する[*8]．

```
> 2*pt;
(3/4 : 15/8 : 1)
> 3*pt;
(431/49 : -12377/343 : 1)
> 2*pt+3*pt;
(14907791/2486929 : 54409047141/3921887033 : 1)
> $1 eq 5*pt;
true
> Height(pt); NaiveHeight(pt);
0.65903205355516536945102769266 6
1.33902066213028345027690308362E-72
```

楕円曲線上の点の和とスカラ倍，高さなども計算できる．

```
> MM,phi:=MinimalModel(E); MM;
Elliptic Curve defined by y^2 + x*y = x^3 - x^2 + 4*x + 4
over Rational Field
> phi;
Elliptic curve isomorphism from: CrvEll: E to CrvEll: MM
Taking (x : y : 1) to (x + 1 : y + 1 : 1)
> phi(E![-1,-3]);
(0 : -2 : 1)
```

[*8] Sage / CoCalc では `Infinity` と出力される．

```
> invphi:=Inverse(phi); invphi;
Elliptic curve isomorphism from: CrvEll: MM to CrvEll: E
Taking (x : y : 1) to (x - 1 : y - 1 : 1)
```

楕円曲線 E の極小モデル E' を求める関数 MinimalModel では，第 2 の出力として E から E' への同型写像を与える．この場合は x 座標，y 座標共に 1 だけ平行移動する写像となり，E 上の点 $(-1,-3)$ が E' 上の点 $(0,-2)$ に写る様子が分かる．また逆写像（E' から E への同型写像）も関数 Inverse を使えば即座に得られる．

```
> E:=EllipticCurve([GF(5)|7,5]); E;
Elliptic Curve defined by y^2 = x^3 + 2*x over GF(5)
> Points(E);
{@ (0 : 1 : 0), (0 : 0 : 1) @}
> Twists(E);
[
    Elliptic Curve defined by y^2 = x^3 + 2*x over GF(5),
    Elliptic Curve defined by y^2 = x^3 + 4*x over GF(5),
    Elliptic Curve defined by y^2 = x^3 + 3*x over GF(5),
    Elliptic Curve defined by y^2 = x^3 + x over GF(5)
]
```

Magma では有限体上の楕円曲線も扱うことができる．基礎体を変更するには EllipticCurve([K|***]); として指定し，K の部分に基礎体を，*** の部分に係数を入れる．有限体上の全ての点を列挙するには関数 Points を用いる．また関数 Twists を用いれば，与えられた曲線の全ての twist を出力できる．2 次 twist に限定したい場合は

```
> QuadraticTwists(E);
[
    Elliptic Curve defined by y^2 = x^3 + 2*x over GF(5),
    Elliptic Curve defined by y^2 = x^3 + 3*x over GF(5)
]
```

とすればよい．

```
> p:=NextPrime(10^9); p;
1000000007
> E:=EllipticCurve([GF(p)|0,1]);
> SEA(E);
```

```
1000000008
> IsSupersingular(E);
true
```

また Magma には，比較的巨大な素数 p に対して $E/\mathbb{F}_p$ の位数を効率的に計算する Schoof-Elkies-Atkin（SEA）アルゴリズムが実装されており，上のように計算ができる．とくにこの楕円曲線は超特異[*9] *supersingular* である．

SEA アルゴリズムの詳細については，本報告集の安田雅哉氏の記事を参照されたい．

5.3 代数体上の楕円曲線

続いて基礎体が代数体（$\mathbb{Q}$ の有限次拡大体）の場合を考えよう．

```
> P<x>:=PolynomialRing(Rationals());
> N<a>:=NumberField(x^2-5);
> N;
Number Field with defining polynomial x^2 - 5 over the
Rational Field
> E:=EllipticCurve([N|1,1,1,-3,1]); E;
Elliptic Curve defined by y^2 + x*y + y = x^3 + x^2 - 3*x + 1
over N
> MordellWeilGroup(E);
Abelian Group isomorphic to Z/15
Defined on 1 generator
Relations:
    15*$.1 = 0
Mapping from: Abelian Group isomorphic to Z/15
Defined on 1 generator
Relations:
    15*$.1 = 0 to Set of points of E with coordinates in N
    given by a rule [no inverse]
true true
```

上は $\mathbb{Q}(\sqrt{5})$ 上の楕円曲線 $y^2+xy+y=x^3+x^2-3x+1$ である（“N|” の部分がないと $\mathbb{Q}$ 上の楕円曲線としてみなされてしまう）．この Mordell-Weil

[*9] q を素数 p のべき，E を $\mathbb{F}_q$ 上の楕円曲線としたとき $q+1-\#E(\mathbb{F}_q)\equiv 0 \pmod{p}$ をみたすような E のこと．

群 $E(\mathbb{Q}(\sqrt{5}))$ は $\mathbb{Z}/15\mathbb{Z}$ に同型である．実は $\mathbb{Q}(\sqrt{5})$ 上の楕円曲線で位数 15 の torsion point をもつものは（同型を除いて）この楕円曲線のみである（なお Mazur の定理から，$\mathbb{Q}$ 上の楕円曲線には位数 15 の torsion point をもつものは存在しない）．

```
> pt3:=Generators(E)[1]; pt3;
(-2*a + 5 : 8*a - 18 : 1)
> 15*pt3;
(0 : 1 : 0)
```

位数 15 の点が $(5-2\sqrt{5}, -18+8\sqrt{5})$ であることが確かめられる．なお射影座標における $(0:1:0)$ は無限遠点 O を表す．

```
> G,phi:=UnitGroup(N); G;
Abelian Group isomorphic to Z/2 + Z
Defined on 2 generators
Relations:
    2*G.1 = 0
> phi;
Mapping from: GrpAb: G to Maximal Order of Equation Order
with defining polynomial x^2 - 5 over its ground order
> u:=N!phi(G.2);
> u;
1/2*(a + 1)
> Norm(u);
-1
```

これは $\mathbb{Q}(\sqrt{5})$ の基本単数を一つ求めるための計算である．$\mathbb{Q}(\sqrt{5})$ の単数群の生成元は 2 つあるが，1 つ目が位数 2，2 つ目が無限位数であるから，非自明な 2 つ目を採用し u に格納している．結果として得られた基本単数は $(1+\sqrt{5})/2$ で，そのノルムは -1 である．

```
> E:=EllipticCurve([N|0,u+1,0,u,0]); E;
Elliptic Curve defined by
y^2 = x^3 + 1/2*(a + 3)*x^2 + 1/2*(a + 1)*x over N
```

u はもちろん $\mathbb{Q}(\sqrt{5})$ の元として扱われているので，楕円曲線の係数に含めることができる．なおこの場合は基礎体を指定せず，単に

```
> E:=EllipticCurve([0,u+1,0,u,0]);
```

と入力すれば，自動的に $\mathbb{Q}(\sqrt{5})$ 上の楕円曲線として認識される．

```
> I:=ideal<MaximalOrder(N)|3>; I;
Principal Ideal
Generator:
    [3, 0]
> IsPrime(I);
true
> Reduction(E,I);
Elliptic Curve defined by y^2 = x^3 + 2*$.1*x^2 + (2*$.1 + 2)*x
over GF(3^2)
Mapping from: CrvEll: E to Elliptic Curve defined by
y^2 = x^3 + 2*$.1*x^2 + (2*$.1 + 2)*x over GF(3^2)
given by a rule [no inverse]
```

楕円曲線の還元には関数 Reduction を用いる．上は単項イデアル (3) での還元であり，MaximalOrder は代数体 N の整環 $\mathcal{O}_N$ を表す．これによって基礎体は $\mathbb{F}_9$ となる．一方，単項イデアル (2) はこの楕円曲線における bad place であるから，還元することができない．

```
> I:=ideal<MaximalOrder(N)|2>;
> Reduction(E,I);

>> Reduction(E,I);
              ^
Runtime error: model should be integral and of good reduction
at the prime
> Conductor(E);
Principal Ideal
Generator:
    [16, 0]
> Factorization($1);
[
    <Principal Prime Ideal
    Generator:
        [2, 0], 4>
]
```

最後に Magma に最近実装された関数 EllipticCurveSearch を紹介しよう．

この関数は Cremona-Thongjunthug による j-不変量を用いた楕円曲線の数え上げ関数である．導手を指定すると，その導手をもつ楕円曲線を走査する．走査範囲の大小はオプション `Effort` で指定する．ここでは `Effort=10` で固定する．まずは導手として norm conductor 2 のものを探す．

```
> I1:=ideal<MaximalOrder(N)|2>; I1;
Principal Ideal
Generator:
    [2, 0]
> SetVerbose("ECSearch",1);
> EllipticCurveSearch(I1,10);
Checking for curves with j-invariant 0 or 1728
Checking Q-rational curves with conductors [ 2, 50 ]

72 candidates for discriminants (up to 6th powers)
Preliminary phase took 0.170s
Effort = 10:
Effort = 10 took 3.180s [memory usage 61M]
[]
```

結果として，そのような楕円曲線は一本も見つからない．というのも，実は $\mathbb{Q}(\sqrt{5})$ 上の楕円曲線のうち，最小の norm conductor は 31 である．

```
> I2:=ideal<MaximalOrder(N)|31>; I2;
Principal Ideal
Generator:
    [31, 0]
> SetVerbose("ECSearch",1);
> ECS:=EllipticCurveSearch(I2,10);
Checking for curves with j-invariant 0 or 1728
Checking Q-rational curves with conductors [ 31, 775 ]

432 candidates for discriminants (up to 6th powers)
Preliminary phase took 0.640s
Effort = 10: Found curve with discriminant -372*a - 1271
(norm 923521) and j = 1/29791*(-102400*a + 10518528)
Coefficients: [0, 1/2*(-a + 1), 1, 2, 1/2*(-a - 3)]
```

```
Effort = 10 took 12.200s [memory usage 95M]
> ECS;
[
    Elliptic Curve defined by y^2 + y = x^3 + 1/2*(-a + 1)*x^2
    + 2*x + 1/2*(-a - 3) over N,
    Elliptic Curve defined by y^2 + y = x^3 + 1/2*(a + 1)*x^2
    + 2*x + 1/2*(a - 3) over N
]
> E:=ECS[1]; Conductor(E);
Principal Ideal
Generator:
    [31, 0]
> E:=ECS[2]; Conductor(E);
Principal Ideal
Generator:
    [31, 0]
```

今回は該当する楕円曲線が 2 本見つかり，確かにどちらも導手が (31) であることが確認できる．

5.4 超楕円曲線

最後の節として，講演では紹介できなかった超楕円曲線の計算について解説する．R を整域とする（今回は簡単のため，$R = \mathbb{Q}$ として計算を進める）．$f(x), h(x)$ を共に R 係数の一変数多項式としたとき

$$y^2 + h(x)y = f(x)$$

で与えられる非特異曲線を超楕円曲線 *hyperelliptic curve* とよぶ．

```
> P<x>:=PolynomialRing(Rationals());
> f:=x^6+3*x^5-4*x+2;
> h:=x;
> C:=HyperellipticCurve([f,h]); C;
Hyperelliptic Curve defined by
y^2 + x*y = x^6 + 3*x^5 - 4*x + 2 over Rational Field
> Genus(C);
2
> Conductor(C);
```

```
1158971546
> Factorization($1);
[ <2, 1>, <579485773, 1> ]
```

C は種数 2 の超楕円曲線であり，bad prime は 2 と 579485773 の 2 つである．

```
> IgusaInvariants(C);
[ -1920, 128878, 177705, -4237683121, -1158971546 ]
```

$f(x)$ の根の情報を用いて定義される井草不変量 *Igusa invariant* の計算も可能である[*10]．

```
> pt:=C![1,1]; pt;
(1 : 1 : 1)
> Involution(pt);
(1 : -2 : 1)
```

楕円曲線の場合と同様，超楕円曲線上の点を扱うことも可能である．なお関数 `Involution` は，入力された点の hyperelliptic involution であり，`-pt` と入力してもよい．

```
> PointsAtInfinity(C);
{@ (1 : -1 : 0), (1 : 1 : 0) @}
```

楕円曲線の場合とは異なり，この場合は無限遠点が 2 つ存在する．

```
> Cp:=ChangeRing(C,GF(37));
> Cp;
Hyperelliptic Curve defined by
y^2 + x*y = x^6 + 3*x^5 + 33*x + 2 over GF(37)
> Points(Cp);
{@ (1 : 1 : 0), (1 : 36 : 0), (1 : 1 : 1), (1 : 35 : 1),
 (2 : 8 : 1), (2 : 27 : 1), (5 : 14 : 1), (5 : 18 : 1),
 (6 : 32 : 1), (6 : 36 : 1), (7 : 32 : 1), (7 : 35 : 1),
 (12 : 7 : 1), (12 : 18 : 1), (13 : 7 : 1), (13 : 17 : 1),
 (14 : 2 : 1), (14 : 21 : 1), (16 : 29 : 1), (17 : 1 : 1),
```

[*10] 楕円曲線における離散対数問題 ECDLP に基いた暗号理論として楕円曲線暗号が知られているが，超楕円曲線を用いた暗号も存在する．その一つとして，井草不変量を用いて超楕円曲線暗号を構成する手法が知られている．cf. 松尾和人（他 3 名）「井草不変量を用いた超楕円曲線暗号の構成について」(On construction of secure hyperelliptic curve cryptosystems using Igusa invariants)，電子情報通信学会論文誌 A (J84-A-8, 2001)，pp.1045-1053.

```
(17 : 19 : 1), (18 : 8 : 1), (18 : 11 : 1), (19 : 20 : 1),
(19 : 35 : 1), (21 : 6 : 1), (21 : 10 : 1), (22 : 18 : 1),
(22 : 34 : 1), (24 : 5 : 1), (24 : 8 : 1), (25 : 5 : 1),
(25 : 7 : 1), (27 : 13 : 1), (27 : 34 : 1), (31 : 7 : 1),
(31 : 36 : 1), (32 : 17 : 1), (32 : 25 : 1), (33 : 13 : 1),
(33 : 28 : 1), (34 : 11 : 1), (34 : 29 : 1), (35 : 5 : 1),
(35 : 34 : 1) @}
```

基礎体を有限体に取り替えることも容易である．この場合は全ての点を列挙できる関数 Points が使える．

```
> Twists(Cp);
[
    Hyperelliptic Curve defined by
    y^2 = 19*x^6 + 10*x^5 + 27*x^4 + 10*x^3 + 24*x^2 + 28*x + 1
    over GF(37),
    Hyperelliptic Curve defined by
    y^2 = x^6 + 20*x^5 + 17*x^4 + 20*x^3 + 11*x^2 + 19*x + 2
    over GF(37)
]
Symmetric group acting on a set of cardinality 2
Order = 2
```

超楕円曲線の種数が 2 または 3 の場合は，曲線の全ての twist を列挙できる関数 Twists が利用できる．種数が 2 の場合は任意の標数で実行できるが，種数が 3 の場合は基礎体の標数が 11 以上であり，かつ $y^2 = f(x)$（$h(x) = 0$）のタイプの曲線にしか適用できない．

```
> J:=Jacobian(C); J;
Jacobian of Hyperelliptic Curve defined by
y^2 + x*y = x^6 + 3*x^5 - 4*x + 2 over Rational Field
> Dimension(J);
2
> TwoSelmerGroup(J);
Abelian Group isomorphic to Z/2 + Z/2 + Z/2
Defined on 3 generators
Relations:
    2*$.1 = 0
    2*$.2 = 0
```

```
    2*$.3 = 0
Mapping from: Abelian Group isomorphic to Z/2 + Z/2 + Z/2
Defined on 3 generators
Relations:
    2*$.1 = 0
    2*$.2 = 0
    2*$.3 = 0 to Univariate Quotient Polynomial Algebra in
    $.1 over Rational Field
with modulus $.1^6 + 12*$.1^5 + 64*$.1^2 - 4096*$.1 + 8192
given by a rule [no inverse]
> RankBound(J);
3
```

超楕円曲線から定まる Jacobian の計算も行える（とくに上では 2-descent アルゴリズムを実行している）．とくに有限体上の超楕円曲線から定まる Jacobian の計算については，数多くの内部関数を備えている．先ほどの $\mathbb{F}_{37}$ 上の超楕円曲線 C_p の Jacobian を使って実験してみたものが次である．

```
> Jp:=Jacobian(Cp); Points(Jp);
{@ (1, 0, 0), (x^2 + 28*x + 5, 11*x + 36, 2), (x^2 + 11*x + 30,
 18*x + 4, 2), (x^2 + 8*x + 16, 12*x + 2, 2), (x^2 + 9*x + 21,
 13*x + 28, 2), (x + 32, x^3, 2), (x^2 + 12*x + 19, 34*x + 1, 2),
 ... (中略) ...
 (x + 32, 36*x^3 + 32, 2), (x^2 + 9*x + 21, 23*x + 9, 2),
 (x^2 + 8*x + 16, 24*x + 35, 2), (x^2 + 11*x + 30, 18*x + 33, 2),
 (x^2 + 28*x + 5, 25*x + 1, 2) @}
> #$1;
1693
> ptj1:=Random(Jp); ptj1;
(x^2 + 15*x + 16, 30*x + 24, 2)
> ptj2:=Random(Jp); ptj2;
(x^2 + 6*x + 6, 23*x + 27, 2)
> 2*ptj1;
(x^2 + 23*x + 26, 28*x + 24, 2)
> ptj1+ptj2;
(x^2 + 29*x + 5, 7*x + 30, 2)
> Order($1);
1693
```

5.5 おわりに

以上の通り，Magma は楕円曲線・超楕円曲線の計算のための多種多様な関数やアルゴリズムを網羅している．これらをうまく組み合わせ，研究用途・教育用途に幅広く活用していただけることを願う．

Magma を論文等で引用する際は，以下の論文を引用する．

- W. Bosma, J. Cannon and C. Playoust, *The Magma algebra system I. The user language*, J. Symbolic Comput. **24** (1997), no. 3–4, pp.235-265, Computational algebra and number theory (London, 1993).

Magma は基本的にソースコードを公開していないため，明らかにおかしな出力などが見られた場合，バグの可能性を否定できない．このような状況に遭遇した場合は，是非 Magma の開発チームに報告をお願いしたい．とくに

- Magma のバージョン，使用 PC の OS 情報
- Magma 起動時に表示される 10 桁の initial seed 番号
- 具体的なバグ（と思われるもの）の情報

を提供していただければ幸いである．

参考文献

楕円曲線の計算に関する文献として，代表的なものをいくつか挙げておく．

- J. H. Silverman and J. Tate, *Rational Points on Elliptic Curve*, 2nd edition, Undergraduate Texts in Mathematics, Springer-Verlag (1992).
  主に $\mathbb{Q}$ 上の楕円曲線に関する基礎事項を扱っている．具体例も豊富に盛り込まれており感覚が掴みやすい．
- J. H. Silverman, *The Arithmetic of Elliptic Curves*, 2nd edition, Graduate Texts in Mathematics **106**, Springer-Verlag (2009).
  Rational Points... よりも高度であるが，各種基礎体上の楕円曲線に関する事項をほぼ網羅している．
- J. Cremona, *Algorithms for Modular Elliptic Curves*, 2nd edition, Cambridge University Press (1997).
  楕円曲線の計算に特化した一冊．著者による楕円曲線の計算プログラム `mwrank` の設計を知ることができる．主に $\mathbb{Q}$ 上および $\mathbb{F}_p$ 上の楕円曲線の計算を扱う．

- N. P. Smart, *The algorithmic resolution of Diophantine equations*, London Mathematical Society Student Text **41** (1998).
  タイトルには Diophantine equations とあるが，楕円曲線上のすべての整数点を求めるために必要な LLL 格子簡約アルゴリズムなどの詳細を知ることができる．代数体上の楕円曲線の計算において有用である．
- J. Cremona and M. Lingham, *Finding all elliptic curves with good reduction outside a given set of primes*, Exp. Math. **16** No.3 (2007), 303-312.
  2017 年より Magma に実装された `EllipticCurveSearch` の基礎となった論文である．
- T. Thongjunthug, *Heights on elliptic curves over number fields, period lattices, and complex elliptic logarithms*, Ph. D. Thesis, The University of Warwick (2011).
  `EllipticCurveSearch` に採用された手法を詳細に述べた博士論文．
  short version は J. Cremona との共著として
- J. Cremona and T. Thongjunthug, *The complex AGM, periods of elliptic curves over $\mathbb{C}$ and complex elliptic logarithms*, J. Number Theory **133**, Issue 8 (2013), 2813-2841.
  として発表された．

Shun'ichi Yokoyama
Faculty of Mathematics, Kyushu University
s-yokoyama@math.kyushu-u.ac.jp