TECC – KASH による楕円曲線の計算*

木田 雅成†
電気通信大学

1 KASH とは何か

KANT [6] は数論の計算, 特に代数的数論の計算をするために開発されたベルリン工科大学の Pohst 教授を中心とするグループによって開発されたソフトウェアである. KASH (KAnt SHell) はその名のとおり, KANT ライブラリへのインターフェイスを与えている. KASH は代数的数論を中心とする分野の数論ソフトとしては, Pari/GP とならんで, 最も大規模で, また最も良く使われているものといって良いであろう.

いろいろな UNIX や Microsoft Windows などで動作する KASH のバイナリファイルが無料で配布されている. ただしソースコードは公開されていない[1].

より詳しいことは KANT/KASH の home page

http://www.math.tu-berlin.de/algebra/

を参照のこと.

2 TECC – Tiny Elliptic Curve Calculator

2.1 TECC とは何か

TECC は KASH のプログラム言語で書かれた楕円曲線計算機で, 有理点などは計算できないが, 任意の代数体上で定義された楕円曲線の reduction や height 関数などが計算できるように作られている.

TECC は論文 [9, 10] における計算を行うために, 書かれた関数群が起源であって, それらをこの研究集会を機会に整理・拡張し, マニュアル等を整備したものである. したがって, TECC の関数を組み合わせることによって, Mordell–Weil 群が

*この原稿は第三回「代数学と計算」研究集会のために用意したスライドに訂正および簡単な加筆を施したものです.

†この研究は文部省科学研究費補助金奨励研究 (A)(No. 11740009) の援助をうけています.

[1]以前は C のライブラリが公開されていた. それ以前には Fortran のライブラリであったことを中村 憲, 山村 健の両氏に研究集会のおりに教えていただいた.

わかっている楕円曲線に対して, [16] の方法にしたがって, 整数点を計算することができることを目標に作られており, 実際それは可能になっている.

有理数体上の楕円曲線の様々な不変量を計算するプログラムは最後のセクションで見るようにたくさんあるが, 一般の代数体上の楕円曲線の極小モデルや reduction などを計算できるものは多くない. また, 二次体に限ればできると標榜しているものでも多くの不具合があることが知られている. したがって, TECC のような特徴をもつ新たな楕円曲線計算機を発表することには意義があると考えられる.

上に述べたような計算を実現するためには代数体の数論の高度の計算が不可欠である. 例えばイデアル類群の生成元の計算, 代数体の素イデアル分解, 代数体上の方程式の因数分解などが可能であることが必要条件である. このことを考慮して, 代数的数論に関する多くの関数を備えた KASH 上のプログラムとして TECC を実現することにした. KASH の高機能な関数群を使用しているために TECC 自体は非常に小さいプログラムにすることができた.

TECC の最新版と, TECC に関する情報は TECC のページ

http://matha.e-one.uec.ac.jp/~kida/TECC.html

から得られる.

以下では, TECC を使うとどういう計算が可能であるかを簡単に概観する. 詳細については, 英文マニュアル [11] をみていただきたい.

なお TECC version 2.1 を動かすには, KASH version 2.2 が必要である.

2.2 初期化

まず TECC の関数を読み込む.

```
kash> Read("tecc.kash");
TECC Version 2.1
Copyright Masanari Kida 1998-1999
```

楕円曲線は必ず `ECInit` 関数で初期化する[2]. この時に定義体を指定する必要がある. 以下プログラム例の中では # 以下はコメントである.

```
kash> O:=OrderMaximal(Z,2,2);; # ℚ(√2)
kash> el:=ECInit([1,2,3,4,5],O);
Elliptic Curve [a1,a2,a3,a4,a6]=[1,2,3,4,5]
 defined over the order: Generating polynomial: x^2 - 2
Discriminant: 8
```

これで, $\mathbb{Q}(\sqrt{2})$ 上の楕円曲線

$$y^2+xy+3y=x^3+2x^2+4x+5$$

[2]楕円曲線を e に代入することはできないことに注意. KASH では e は予約語である.

を定義したことになる. ECInit 関数は内部で基本的な不変量を計算しているが, それらはすべて $\mathbb{Q}(\sqrt{2})$ の元として扱われる.

楕円曲線上の点は長さ 2 のリスト [a,b] であらわす. ただし, 単位元は [0] であらわす.

```
kash> E:=ECInit([1,2,3,4,5],Z);;
kash> ECPtIsOnCurve(E,[1,2]); # (1,2)∈E(ℚ)?
true
kash> ECPtIsOnCurve(E,[0]);  # ∞∈E(ℚ)?
true
```

基本的な不変量は .hogehoge のかたちで呼び出すことができる.

```
kash> E.b4; # b4
11
kash> E.jinv; # j-invariant
6128487 / 10351
```

2.3 群演算

もちろん, 群演算もできる.

```
kash> P1:=[1,2];;P2:=[ -103/64, -233/512 ];;
kash> ECPtIsOnCurve(E,P2); # P2∈E(ℚ)?
true
kash> ECAddPt(E,P1,P2); #  P1+P2
[ 12145/27889, -22841754/4657463 ]
kash> nP1:=ECInvPt(E,P1); #  -P1
[ 1, -6 ]
kash> ECAddPt(E,nP1,P1);  #  -P1+P1
[ 0 ]
kash> ECMultPt(E,P1,3); #  3P1
[ 12145/27889, -22841754/4657463 ]
kash> ECHalfPt(E,P2);
[ [ 1, 2 ] ]
```

ECHalfPt は与えられた点 P に対して, $P = 2Q$ をみたす点 Q のリストを返す. この計算もはじめに指定した楕円曲線の定義体の中で行われる. この計算には Washington のアルゴリズム [18, Proposition 4] を使っている.

2.4 Minimal model, reduction

TECC ではすべての計算に global minimal model の存在を仮定していないが, global minimal model を使うといろいろな計算が早くできることがあるので, これを求める関数は用意してある.

```
kash> O:=OrderMaximal(Z,2,6);; # ℚ(√6)
kash> eps:=OrderUnitsFund(O)[1];; # ℚ(√6) の基本単数
kash> el:=ECInit([0,-2*(eps-1),0,4*eps,0],O);
Elliptic Curve [a1,a2,a3,a4,a6]=[0,[-8, -4],0,[20, 8],0]
 defined over the order:
Generating polynomial: x^2 - 6
Discriminant: 24
Regulator: 2.292431669561177687800787311348015431621868240016
Fundamental unit:
[5, 2]
kash> Factor(el.disc); # 判別式の素イデアル分解
[ [ <2, [0, 1]>, 24 ] ]
```

したがって, 上の楕円曲線は素イデアル `<2, [0, 1]>` では minimal ではないかもしれない. Global minimal model があるかどうかを調べてみよう.

```
kash> eg:=ECGlobalMinimalModel(el);
[ [ Elliptic Curve [a1,a2,a3,a4,a6]=[[0, 1],1,[1, 1],[-203, 82],[1215, -497]]
         defined over
        the order whose Generating polynomial: x^2 - 6
        Discriminant: 24
        Regulator: 2.292431669561177687800787311348015431621868240016
        Fundamental unit:
        [5, 2]
        , [ [10, 4], [166, 68], [12, 5], [6692, 2732] ] ],
  [ Elliptic Curve [a1,a2,a3,a4,a6]=[0,[1, -1],0,[-31, -14],[-75, -31]]
         defined over
        the order whose Generating polynomial: x^2 - 6
        Discriminant: 24
        Regulator: 2.292431669561177687800787311348015431621868240016
        Fundamental unit:
        [5, 2]
        , [ 1, [3, 1], 0, 0 ] ] ]
```

関数 `ECGlobalMinimalModel` は Laska–Kraus–Connell のアルゴリズム [3, Chapter 5] を使って, 一般に複数の global minimal model の候補を返す. それは, global minimal model が存在しない場合でも quasi global minimal model や local minimal model があると便利な場合があるからである. 実際に global minimal model が存在するかを調べるには, その Weierstrass class を計算してやればよい.

```
kash> ECWeierstrassClass(eg[1][1]);
[ <1>, 1 ]  # [Weierstrass divisor, Weierstrass class]
```

となって, `eg[1][1]` が global minimal model であることがわかる. これは $\mathbb{Q}(\sqrt{6})$ の類数が 1 であるから当然である.

以下に述べる Tate のアルゴリズムを使っても global minimal model の候補は見つかるが, global minimal model を見つけるだけならば, 上の関数のほうが高速である.

Reduction は `ECGlobalReduction` で調べる. この関数は Tate のアルゴリズム [14, IV.9] の実装である.

次の例は第二回「代数学と計算」研究集会の梅垣敦紀氏[3] の講演 [17] ([7] も参照のこと) で与えられたものである.

$$\begin{aligned} E_1 : y^2 = x^3 &+ \left(9 + \frac{\sqrt{6}}{2} + \frac{\sqrt{6}\sqrt{29}}{2}\right)x^2 \\ &(-383506419653 - 156534506597\sqrt{6} \\ &\quad + 71201118525\sqrt{29} + 29073539873\sqrt{6}\sqrt{29})x \\ &- 182798829223792711 - 74627160360067580\sqrt{6} \\ &\quad 33944822557919841\sqrt{29} + 13857943481193026\sqrt{6}\sqrt{29} \end{aligned}$$

判別式は

$$\Delta(E_1) = \mathfrak{p}_2^{12} \cdot \mathfrak{p}_5^2 \cdot (\mathfrak{p}_5^{\sigma})^{11} \cdot \mathfrak{p}_5^{\tau} \cdot (\mathfrak{p}_5^{\sigma\tau})^{22}$$

である. 詳しい記号については上記論文を参照のこと.

```
kash> O:=Order(Z,2,6);;x:=Elt(O,[0,1]);; #  x = √6
kash> x^2;
6
kash> O:=OrderMaximal(O);;x:=EltMove(x,O);;
kash> OO:=Order(O,2,29);;y:=Elt(OO,[0,1]);; #  y = √29
kash> y^2;
29
kash> OO:=OrderMaximal(OrderAbs(OO));; # ℚ(√6,√29)
kash> x:=EltMove(x,OO);;y:=EltMove(y,OO);;
kash> E1:=ECInit([0,9+x/2+x*y/2,0,-383506419653-156534506597*x
> +71201118525*y+29073539873*x*y,-182798829223792711
> -74627160360067580*x+33944822557919841*y+13857943481193026*x*y],OO);;
kash> L:=Factor(E1.disc);
[ <5, [1, 1, 0, 0]>, <5, [2, 1, 0, 0]>, <5, [3, 1, 0, 0]>, <5, [4, 1, 0, 0]>,
  <
    [2 1 0 0]
    [0 1 0 0]
    [0 0 2 1]
    [0 0 0 1]
    >
     ]
kash> List(L{[1..Length(L)]}[1],x->IdealIsPrincipal(x));
[ [1, 5, 0, -3], [8, -5, -1, 3], [-5, -4, 1, 3], [12, -11, -1, 7],
  [-11, 11, 1, -7] ] # [p5^στ,p5,p5^τ,p5^σ,p2]
kash> Time();;
kash> ECGlobalReduction(E1);
[ [ <[-33806020, 34634920, 2775960, -21748700]>, # 導手
      [ 1, [9, 0, 1, 1], -1, [-230, -199, 63, 169] ] ], # ここまで
が global data
```

[3] 梅垣氏はこの講演で reduction を求める Tate のアルゴリズムの Pari への実装を紹介している.

```
  [ [ <[-11, 11, 1, -7]>, -2, 4, 1, 9, 12,
          [ 1, [0, 0, 1, 1], -1, [-221, -199, 64, 170] ] ], # II* at p_2
      [ <[1, 5, 0, -3]>, 26, 1, 2, 22, 22, [ 1, 2, 0, 0 ] ], # I_22 at p_5^σ
      [ <[8, -5, -1, 3]>, 6, 1, 2, 2, 2, [ 1, 2, 0, 0 ] ], # I_2 at p_5
      [ <[-5, -4, 1, 3]>, 5, 1, 1, 1, 1, [ 1, 2, 0, 0 ] ], # I_1 at p_5^στ
      [ <[12, -11, -1, 7]>, 15, 1, 11, 11, 11, [ 1, 3, 0, 0 ] ] ] ] # I_11 at p_5^τ
Time: 1340 ms
```

となって, 梅垣氏の計算とピッタリあう. `ECGlobalReduction` の返す値についてはマニュアルを参照のこと.

KASH で書いてあるので, 計算が遅いのではないかと思われるかもしれないが, 上のような例を計算するのであれば実用の範囲内であると考えられる[4] .

2.5 Height functions

TECC は canonical height も計算できる. アルゴリズムは Silverman [13] によるものである. ただし global minimal model の存在を仮定していないので, 少し修正が必要である.

以下の例は [12] で構成された階数が高い楕円曲線である.

```
kash> el:=ECInit([1,1,1,-215843772422443922015169952702159835,
-194743612777871519472559614354590541515017922413205 35],Z);
Elliptic Curve [a1,a2,a3,a4,a6]=[1,1,1,-215843772422443922015169952702159835,
-194743612777871519472559614354590541515017922413205 35]
defined over Integer Ring
```

この楕円曲線には, 21 個の独立な点がのっているが, そのうちのはじめのものについて canonical height を計算する.

```
kash> ECCanonicalHeight(el,[8008430088893400659 33/16,
                        22662214190910903990783584765347/64]);
19.463227693334260602258825
Time: 41 s
```

実は この値は [15] で計算されている値とは一致しない. Pari/GP の計算値とは一致する. 長尾氏によれば TECC, Pari/GP での計算値のほうが正しそうであるとのことである.

浮動小数点演算は遅いので, `Prec` および `OrderPrec` を適切に設定する必要がある. Height の関連では Neron–Tate paring を計算したり (`ECNeronTateParing`), elliptic regulator を計算したり (`ECRegulator`) もできるが, 上の楕円曲線について, TECC を使って 21 個の点の独立性を証明しようなどとは考えない方が無難である.

[4] 計算時間は Pentium Pro 200MHz の CPU をもつ Linux マシン上で計測したものである.

2.6 Elliptic curves over $\mathbb{C}$

代数体を $\mathbb{C}$ に埋め込むことによって, 代数体上の楕円曲線 E を $\mathbb{C}$ 上の楕円曲線と見なして, その周期や楕円対数などを計算することができる. これらの関数の計算結果は代数体の $\mathbb{C}$ への埋め込みに対応して`OrderDegAbs(E.order)` 個のリストとして与えられる.

以下の例に見る楕円対数を与える関数 `ECElog` と `ECztoPoint` は互いに逆関数の関係にある.

$$E(\mathbb{C}) \underset{\texttt{ECztoPoint}}{\overset{\texttt{ECElog}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\simeq}}}} \mathbb{C}/L$$

```
kash> el:=ECInit([0,0,0,-112,400],Z);
Elliptic Curve [a1,a2,a3,a4,a6]=[0,0,0,-112,400]
defined over Integer Ring

kash> ECElog(el,[-4,-28]);
[ 0.84688142751629308056172158038439 + 0.37013706706035374832683277765*i ]
kash> ECztoPoint(el,last[1]);
[ [ -4.00000000000000000000000000006 + 0.00000000000000000000000000022*i,
      -27.99999999999999999999999999971 + 0.00000000000000000000000000455*i ] ]
```

残念ながら, `ECElog` だけは基礎体などに制限がある. つまり総実代数体上で定義された楕円曲線上の, 実数に埋め込める点しか計算できない. `ECElog` は Pari の実装とほぼ同じであるが, このままでは一般の代数体上の楕円曲線には適用できないのである. いろいろ文献に当たってみたが, 適当なアルゴリズムを見つけることができなかった[5]. ご存じであればお教え願いたい.

2.7 虚数乗法

虚数乗法に関することもできる.

```
kash> O:=Order(HilbertClassPoly(-15));
# 判別式 −15 の Hilbert 類多項式の生成する体 (= ℚ(√5))
Generating polynomial: x^2 + 191025*x - 121287375
kash> t:=Elt(O,[0,1]);; # その Hilbert 類多項式の根
kash> OM:=OrderMaximal(O);;tm:=EltMove(t,OM);;
kash> el:=ECGenericCurve(tm); # j-invariant=tm であるような楕円曲線
Elliptic Curve [a1,a2,a3,a4,a6]=[0,0,0,
     [-30731133510, 49727382705],
          [3962170100466270, -6410928347984160]]
defined over the order:
   F[1]
```

[5] アルゴリズム中では複素数の平方根を計算する必要が出てくるのであるが, その平方根の選び方について何ものべていない文献が多い.

```
     |
    F[2]
   /
  /
Q
F  [ 1]     Given by transformation matrix
F  [ 2]     x^2 + 191025*x - 121287375
Discriminant: 5

# その楕円曲線は当然ながら虚数乗法を判別式 −15 の整環にもつ.
kash> ECHasCM(el);
-15
```

2.8 KASH で Cremona の data を使う

TECC にはおまけとして, `cremona.pl` という Perl スクリプトがついてくる. このプログラムによって Cremona の楕円曲線の data[6] から KASH で読める `cremona_data.kash` が作られる. このファイルには一つの大きいリスト `Cremona_Data` が定義されていて, それは導手が 1000 までの楕円曲線の data を次のようなかたちで保持している.

```
[conductor,"name",[a1,a2,a3,a4,a5],rank,number of torsion,
                                  [generator1],[generators2]..]
```

これを KASH のリスト操作関数を使うことにより, mini database として使おうというのが目的である.

```
kash> Read("cremona_data.kash");
kash> Filtered(Cremona_Data,x->x[4]=2); # 階数 2 の楕円曲線のリスト
[ [ 389, "A1", [ 0, 1, 1, -2, 0 ], 2, 1, [ -1, 1 ], [ 0, 0 ] ],
  [ 433, "A1", [ 1, 0, 0, 0, 1 ], 2, 1, [ -1, 1 ], [ 0, 1 ] ],
    中略
  [ 997, "B1", [ 0, -1, 1, -5, -3 ], 2, 1, [ -1, 0 ], [ 5, 8 ] ],
  [ 997, "C1", [ 0, -1, 1, -24, 54 ], 2, 1, [ 6, -10 ], [ 1, 5 ] ] ]
```

Cremona の本 [4] の楕円曲線 “37A1” を定義する.

```
kash> ECInit(Filtered(Cremona_Data,x->(x[1]=37 and x[2]="A1"))[1][3],Z);
Elliptic Curve [a1,a2,a3,a4,a6]=[0,0,1,-1,0]
defined over Integer Ring

kash> Maximum(List(Cremona_Data,x->x[5])); # 最大の捩じれ位数
16
kash> Filtered(Cremona_Data,t->(t[5]=16)); # それを与える曲線の data
[ [ 210, "E2", [ 1, 0, 0, -1070, 7812 ], 0, 16 ] ]
```

TECC とあわせて使えば, いろいろな実験に役に立つであろう.

[6] http://www.maths.nott.ac.uk/personal/jec/ftp/data/

2.9 TECC にできないこと

もちろん TECC は万能ではない[7]. できないことはたくさんあるが主なものをあげると次のようになる.

有理点の計算 有理数体上の楕円曲線の有理点を計算できるソフトはたくさんあるし, 一方で一般の代数体上では [5] 等の研究があるにせよ, まだ難しい部分がある. また KASH で計算するにはクリアすべき問題が多いと感じられる.

有限体上の楕円曲線の計算 これも特別の工夫が要る分野であるし, 優れた実装も多数ある.

大きな代数体上の楕円曲線の計算 これは KASH の限界による. 大きな代数体を作ることに限界があるし, たとえ作れたにしても, そのうえでイデアルの計算などを行うのは難しいであろう.

係数の大きい楕円曲線の計算 TECC では内部で係数や判別式の因数分解などを頻繁に使っている. したがって, それらが大きくなれば, 計算できない不変量が増えてくる.

3 KASH を使うときに注意すること

この節では, KASH を使う上で注意したほうが良いことをいくつか書いておく.

3.1 Prec と OrderPrec

`Prec` と `OrderPrec` は適切に設定しなくてはならない. この値が低すぎると, 次のようなことがおこり, 正しい答えは得られない.

```
kash> Prec(10);;OrderPrec(10);;
kash> O:=OrderMaximal(Order(Z,2,-74));; # ℚ(√6)
kash> OrderClassGroup(O); # 類数は 10
[ 10, [ 10 ] ]
kash> OrderClassGroupCyclicFactors(O);
[ [ <18, [4, -1]>, 10 ] ]
kash> P:=OrderClassGroupCyclicFactors(O)[1][1]; # 類群の生成元
<18, [4, -1]>
kash> for k in [1..10] do
> Print(IdealIsPrincipal(P^k),"\t"); # P^k が単項かどうか調べてみると
>  od;
false   false   false   false   false   false   false   false   false   false
```

[7] だから tiny なのだ.

もちろんこれは, `Prec` と `OrderPrec` をより大きい値にすれば解決するが, その場合, 計算は遅くなる.

なお version 2.2 の KASH には, この二つの値を同じ値にしておかないと, ある種の計算ができない不具合がある. マニュアルの Chapter 1 を参考のこと.

3.2 KASH はだんだん遅くなる

KASH は計算をさせていくうちに少しずつ遅くなる.

```
kash> OrderClassGroup(OrderMaximal(Z,2,95));
[ 2, [ 2 ] ]
Time: 1140 ms      # ★
kash> for n in Filtered([2..100],IntIsSquareFree) do # 何か仕事をさせる
> OrderClassGroup(OrderMaximal(Z,2,n));
> od;
kash> OrderClassGroup(OrderMaximal(Z,2,95));
[ 2, [ 2 ] ]
Time: 1230 ms     # ★
kash> for n in Filtered([200..400],IntIsSquareFree) do # 何か仕事をさせる
> OrderClassGroup(OrderMaximal(Z,2,n));
> od;
kash> OrderClassGroup(OrderMaximal(Z,2,95));
[ 2, [ 2 ] ]
Time: 1520 ms     # ★
```

★ をつけたところを見ると, 同じ計算をやらせても少しずつ遅くなっていくのがわかるであろう. Maple の restart コマンドのようなコマンドが KASH にもあれば良いとつくづく思う.

3.3 その他

KASH では $\mathbb{Z}$ と一般の代数体を分離して扱わなければならない場合が非常に多い. 例えば `EltCon` 関数で引数に有理整数を与えるとエラーになってしまうことなど. TECC の中でも $\mathbb{Z}$ の場合と一般の整数環の場合で場合わけをして, 無駄な処理をしている部分がたくさんある.

また複素数をあつかうプログラムは書きにくい. それに浮動小数点演算があまり洗練されていないように感じる[8]. 講演時にはいろいろな不具合[9]の実例もあげたが, 今後修正されることに期待してこの報告には書かないことにする[10].

[8] 福田 隆氏の TC を見習ってほしい.

[9] KASH の不具合は kant@math.tu-berlin.de に報告すればよい.

[10] プログラムに不具合があるのは, 仕方がないことであるが, バグ情報や, 修正情報の公開が進んでほしいと筆者は切に願っている. Pari/GP の open な開発がとても羨ましく感じるこの頃である.

4 楕円曲線の計算のできるソフトウェアの紹介

楕円曲線の計算のできるソフトウェアを, 筆者が使ったことのあるものを中心に, いくつか紹介する. 特徴は主観的なものであるのであることをあらかじめお断りしておく.

4.1 Pari/GP version 2.0.17 beta

Home Page http://hasse.mathematik.tu-muenchen.de/ntsw/pari/Welcome

形態 C ライブラリとインタプリタ

ソース あり

稼動 OS UNIX

特徴など *L* 関数の関連が充実.

4.2 SIMATH version 4.2.10

Home Page http://emmy.math.uni-sb.de/ simath/

形態 C ライブラリとインタプリタ

ソース あり

稼動 OS UNIX

特徴など 二次体上の楕円曲線を扱える関数がいくつかある (不具合も多いので注意が必要). Mordell–Weil rank を求める関数がいろいろなアルゴリズムで実装されている.

関連論文など [8]

4.3 Apecs version 4.51

FTP ftp://www.math.mcgill.ca/pub/apecs

形態 Maple 上のプログラム

ソース あり

稼動 OS Maple が稼動する OS

特徴など 有理点の計算が充実. 任意の体上で isogeny の計算もできる. README file では, 最新版は version 5.41 となっているが, どこにあるんだろうか.

関連論文など [3]

4.4 MAGMA version 2.4

Home Page http://www.maths.usyd.edu.au:8000/u/magma/

形態 インタプリタ

ソース なし

稼動 OS UNIX, Microsoft Windows をはじめ様々な OS で動く.

特徴など Mordell–Weil 群の計算をはじめ, isogeny や isomorphism の計算など, 様々な関数が網羅的にそろっている. Mordell–Weil 群の計算は, 他のソフトに比べて速い. 楕円曲線の database ももっている. ここであげた中ではもっとも大規模なシステム. ただし有料である.

関連論文など [1]

4.5 LiDIA version 1.4 alpha 2

Home Page http://www.informatik.th-darmstadt.de/TI/LiDIA/

形態 C++ ライブラリ

ソース あり

稼動 OS UNIX, Windows NT, Macintosh

特徴など 現在開発中のバージョン. マニュアルからの情報では MAGMA の大きな subset になりそうな感じである.

最後にプログラムを書く際に参考にした文献として [4] と [2] をあげておく. より詳細な文献については [11] の文献表を参照していただきたい.

参考文献

[1] W. Bosma, J. Cannon, and C. Playoust, *The Magma algebra system. I. The user language*, J. Symbolic Comput. **24** (1997), no. 3-4, 235–265, Computational algebra and number theory (London, 1993). 4.4

[2] H. Cohen, *A course in computational algebraic number theory*, Springer-Verlag, Berlin, 1993. The errata are available from ftp://megrez.math.u-bordeaux.fr/pub/cohenbook/ 4.5

[3] I. Connell, *Elliptic curve handbook*, McGill University, Montreal, 1996, available from ftp://ftp.math.mcgill.ca/pub/ECH1/. 2.4, 4.3

[4] J. E. Cremona, *Algorithms for modular elliptic curves*, second ed., Cambridge University Press, Cambridge, 1997. 6, 4.5

[5] J. E. Cremona and P. Serf, *Computing the rank of elliptic curves over real quadratic number fields of class number* 1, Math. Comp. **68** (1999), no. 227, 1187–1200. 7

[6] M. Daberkow, C. Fieker, J. Klüners, M. Pohst, K. Roegner, M. Schörnig, and K. Wildanger, *KANT V4*, J. Symbolic Comput. **24** (1997), no. 3-4, 267–283, Computational algebra and number theory (London, 1993). 1

[7] T. Hibino and A. Umegaki, *Families of elliptic $\mathbb{Q}$-curves defined over number fields with large degrees*, Proc. Japan Acad. Ser. A Math. Sci. **74** (1998), no. 1, 20–24. 3

[8] C. Hollinger and P. Serf, *SIMATH—a computer algebra system*, Computational number theory (Debrecen, 1989), de Gruyter, Berlin, 1991, pp. 331–342. 4.2

[9] M. Kida, *Computing elliptic curves having good reduction everywhere over quadratic fields*, (1998), Preprint. 2.1

[10] ———, *Good reduction of elliptic curves over imaginary quadratic fields*, (1999), Prepirnt. 2.1

[11] ———, *TECC manual version 2.2*, The University of Electro-Communications, November 1999. 2.1, 4.5

[12] K.-i. Nagao and T. Kouya, *An example of elliptic curve over $\mathbb{Q}$ with rank ≥ 21*, Proc. Japan Acad. Ser. A Math. Sci. **70** (1994), no. 4, 104–105. 2.5

[13] J. H. Silverman, *Computing heights on elliptic curves*, Math. Comp. **51** (1988), no. 183, 339–358. 2.5

[14] ———, *Advanced topics in the arithmetic of elliptic curves*, Springer-Verlag, New York, 1994. The errata are available from http://www.math.brown.edu/ jhs/ 2.4

[15] ———, *Computing canonical heights with little (or no) factorization*, Math. Comp. **66** (1997), no. 218, 787–805. 2.5

[16] N. P. Smart and N. M. Stephens, *Integral points on elliptic curves over number fields*, Math. Proc. Cambridge Philos. Soc. **122** (1997), no. 1, 9–16. 2.1

[17] A. Umegaki, *Computing elliptic curves over number fields (in japanese)*, Proceeding of the 2nd Symposium on Algebra and Computation, Tokyo Metropolitan University, 1997, available from ftp://tnt.math.metro-u.ac.jp/pub/ac97/PROCEEDINGS/umegaki/. 3

[18] L. C. Washington, *Class numbers of the simplest cubic fields*, Math. Comp. **48** (1987), no. 177, 371–384. 2.3

きだ まさなり
〒 182-8585 調布市調布が丘 1-5-1
電気通信大学 数学教室
e-mail: kida@matha.e-one.uec.ac.jp

(講演 1999 年 10 月 20 日)
(提出 1999 年 11 月 30 日)