Pari-GP 入門編

木村巌

（富山大学 理工学研究部）

2006 年 7 月 12 日

今日やること

1. Pari-gp の紹介
2. Microsoft Windows へのインストールの過程
3. 起動と終了
4. Pari-gp に組み込みの型、単純な計算
5. 2 次体の計算
6. 一般次数の代数体の計算
7. Galois 拡大に関する計算
8. ちょっとしたプログラミング

プログラミングに関しては、ほんのさわりの部分のみ．コマンドラインでできること $+\varepsilon$.

Pari-gp とは？

数論に特化した計算パッケージ.

Pari C 言語などでプログラムを書く際に用いるライブラリ

gp 対話的に計算するためのソフトウエア

GP gp が理解する言語

開発陣 K. Belabas を中心としたフランスの計算機数論グループ

users manual（200 頁余)、tutorial（50 頁弱)、リファレンスカードなどが添付されている

今日の話は、tutorial を参考にしています．一読をお薦めします．

こんな関数があるのか、というのを知るために、リファレンスカードも手元にあると良い．

ちなみに GAP のマニュアルは、インデックスだけで 50 ページ．

Pari-gp とは？(続)

入手先 `http://pari.math.u-bordeaux.fr/`

ライセンス GNU Public License に基づくフリーソフト

動作環境 Linux, FreeBSD 他の Unix clone, Apple MacOS X, Microsoft Windows

Pari-gp についてよく受ける質問

Q: ～は計算できるか？

A: 場合による：

- 組み込み関数を呼び出すだけでできる
- 組み込み関数をいくつか組み合わせるとできる
- 既知のアルゴリズムを実装するとできる
- アルゴリズムを考案し、それを実装するとできる

Pari-gp についてよく受ける質問（続）

Q: ～はどのくらいの範囲まで計算できるか？

A: マシンの性能、メモリ容量などによる

- gp を起動した時、割り当てるメモリサイズ（デフォルトでは 4MB）
- 同じく、起動した時に計算する素数のリスト（デフォルトでは 500,000 まで）
- 使えるデータ（既存の数表など）を使えるか
- 待ち時間（忍耐力）

Pari-gp についてよく受ける質問（続々）

Q: C 言語でプログラムを書いたほうが、やはり gp でスクリプトを書くよりも速いのか？

A:

- 組み込み関数をいくつか呼び出す程度では、大差ない
- ループを回す場合、gp の for() などは非常に遅い
- C でプログラムを書いた場合、動くようになるまでの時間
- gp2C トランスレータがある（gp のスクリプトを C に変換してくれる）

Pari-gp についてよく受ける質問

Q: 数学的な概念／量を、どのように計算機に載せているのか？

A: マニュアルに記載あり

（有限 Abel 群……Smith Normal Form, 自由 Abel 群……Hermite Normal Form, 指標、イデアル、イデール等々）

いくつかは後述.

Pari-gp についてよく受ける質問

Q: 〜はどうやって計算しているのか？

A: 組み込み関数に実装されているアルゴリズムは、ほとんどが
H. Cohen の二冊の本に解説されている（はず）

- GTM 138, GTM 193

Pari-gp についてよく受ける質問

Q: ～の計算結果は正しいのか？何を仮定して正しいのか？

A: 状況による：

- 無条件に正しい
- GRH を仮定して正しい
- GRH よりも強い Heuristic を仮定して正しい

これも、2 次体、代数体に関する部分については後述.

Microsoft Windows へのインストール

1. `http://pari.math.u-bordeaux.fr/downloads` から，Microsoft Windows 用のコンパイル済みバイナリ（インストーラ付き）を入手．最新安定版は `Pari-2-3-0.exe`.
2. ダブルクリックでインストーラが起動する.

gp の基本概念

組み込み型： 有理整数、有理整数環の剰余環の元、有理数、実数、p 進数、多項式（多変数、係数も任意）、形式巾級数（同）、……

組み込みのデータ型（コンテナ）： （縦・横）ベクトル、行列（= ベクトルのベクトル）

構造体とか、ハッシュのような、より高級なデータ構造はなし．ポインタもなし．複雑なデータ構造を構成するのは難しい.

Pari ライブラリを使用できるプログラミング言語としては、Perl (Math::Pari), Python, Lisp (CLISP) などがある．また、他の数式処理系に組み込まれている例もある（Risa/Asir）.

gp の基本概念（続）

- **exact な型**：有理整数、有理整数の剰余環の元、有理数、それらを係数とする多項式・有理式……
- 基本的に無限多倍長（最近のバージョンは、多倍長計算に GMP を使うこともできる）
- **それ以外**：「精度」の概念がある．実数、p 進数、巾級数など.
- 実数のデフォルトの 10 進での桁数は、28. 巾級数のデフォルトの精度は 16 項まで.

実数の精度の変更は \p、級数の精度の変更は \ps. 例えば、gp のプロンプトに

```
?  \p 56
```

と入力すると、実数の精度が 56 桁になる.

gp の基本概念（続）

横ベクトル：`[1,2,3]`

縦ベクトル: `[1,2,3]~`

行列: `[1,2,3;4,5,6;7,8,9]` $= \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

行列の演算は*: `[1,2,3]*[1,2,3]~ = 14`

行列に関するさまざまな関数. 行列式 `matdet()`, ランク `matrank()`, 固有ベクトル `mateigen()`, 特性多項式 `charpoly()`.

gp の基本概念（続）

変数には型はない.

```
?  a = 1;
?  a = [1, 2, 3];
?  a[1]
1
```

最初に整数を，次に整数のベクトルを代入している.

初等超越関数

sin(): 実数だけでなく，複素数や p 進数も可能.

```
gp> sin(I)
time = 0 ms.
%12 = 0.E-28 + 1.1752011936438014568823818 51*I
```

真値は $(e^{-1} - e)/2i = 1.1752011936438014568823818 51 * I.$

素因数分解など

`factorint()`: 有理整数を MPQS, ECM, ρ, etc... で素因数分解する.

`factor()`: 1 変数多項式（係数はいろいろ）を因子分解する.

`fermat(n)=2^(2^n)+1`.
`sizedigit(n)` で整数 `n` の 10 進桁数が分かる.
debugging level を 3 まで挙げると、因数分解の過程の報告がある：`\g 3`

二次体の計算のデモ

`quad` 一族、`qfb` 一族を使う

`quaddisc(x)` $\mathbf{Q}(\sqrt{x})$ の判別式を返す

`qfbclassno(x)` $\mathbf{Q}(\sqrt{x})$ の類数を返す（Shanks のアルゴリズム）．但し、実装の制限の為、稀に正しくない結果を返すことがある

`quadclassunit(x)` $\mathbf{Q}(\sqrt{x})$ のイデアル類群、単数基準を返す

qfbclassno(x) のデモ

```
?  qfbclassno (-3299)
%10 = 27
qfbclassno (-3299,1)                    ……Euler 積を使う
%11 = 27
```

類数は分かった．類群の構造は？

二次形式の計算もできる：`Qfb(a, b, c)` で、$ax^2+bxy+cy^2$. 合成や簡約も可能.

quadclassunit() のデモ

`quadclassunit(x,{flag}, {tech=[]})` $\mathbf{Q}(\sqrt{x})$ のイデアル類群、単数基準を返す

類群の計算は、Buchmann-McCurley の sub-exponential アルゴリズム.

$D < -10^{25}$, $D > 10^{10}$ に使うと良い（それ以外では `qfbclassno()`）

```
?  quadclassunit(-10^25-3)
%20 = [491852207132, [245926103566, 2], [Qfb(7, 1,
357142857142857142857143), Qfb(13, 13,
192307692307692307692311)], 1]
```

結果は順に、「類数、類群の巡回群の直和としての表示、各巡回成分の生成元（二次形式として)、単数基準」.

quadclassunit(x) の詳細・結果の正当性

flag=1 で $D > 0$ なら狭義類数を計算.

tech=$[c_1, c_2]$ は、計算時間と、必要なメモリ総量を決めるパラメタ.
$c_1, c_2 > 0$ の実数.

$c_2 = c$ なら、最短時間で結果を返す．$0.1 \leq c \leq 2.0$ ぐらい.

$c = 6$ なら、**GRH の元で**正しい結果を返す.

GRH を仮定したくない場合、次の `bnf...` 一族を使う.

```
?  quadclassunit(-3299,,[6,6])
%21 = [27, [9, 3], [Qfb(3, 1, 275), Qfb(23, -17, 39)], 1]
```

GRH を仮定して厳密.

一般の代数体

nf 一族、bnf 一族を使う（big number field/Buchman's number field）

$x^2 + 3299 = 0$ が定義する 2 次体 $\mathbf{Q}(\sqrt{-3299})$ を考えよう.

```
?  T = x^2 + 3299;
?  bnf = bnfinit(T);
?  bnf.clgp
%69 [27, [9, 3], [[3, 2; 0, 1], [23, 8; 0, 1]]]
```

bnf.no=類数=27，bnf.cyc=類群 $\cong \mathbf{Z}/9\mathbf{Z} \oplus \mathbf{Z}/3\mathbf{Z}$, bnf.gen=各列がイデアルの基底を表す

bnfinit() の結果の正当性

bnfcertify() : bnfinit() の返り値を取って，結果が**無条件で正しければ**1 を返す．そうでなければエラーメッセージ，もしくは稀に無限ループ．

```
?  bnf = bnfinit(T);
?  bnfcertify(bnf)
%10 = 1
```

2 次体の Hilbert 類体の計算

quadhilbert(D): 2 次体 $\mathbf{Q}(\sqrt{D})$ の Hilbert 類体の $\mathbf{Q}(\sqrt{D})$ 上の定義多項式を返す.

```
?quadhilbert(-3299)
time = 33 ms.
%11 = x^27 - 125*x^26 + 411202*x^25 - 5273842*x^24 +
5927663*x^23 + 514861*x^22 + 756180179*x^21 -
533202511*x^20 + 3726905423*x^19 - 4436877539*x^18 +
1430746075*x^17 - 1211962575*x^16 - 2629273159*x^15 +
4851196931*x^14 - 690886193*x^13 + 208639789*x^12 -
104541139*x^11 - 1178105817*x^10 - 839425855*x^9 +
691789163*x^8 + 554710685*x^7 - 15194969*x^6 - 9973495*x^5
+ 11421563*x^4 - 2710350*x^3 + 347998*x^2 + 259*x + 1
```

一般の代数体（続）

monic な $\mathbf{Z}$ 係数多項式 T について，`nfinit(T)`, もしくは `bnfinit(T)`.
結果は T の根の一つ θ が $\mathbf{Q}$ 上生成する代数体 $\mathbf{Q}(\theta)$ の様々なデータ.

$x^4 + 24x^2 + 585x + 1791 = 0$ が定義する代数体 K を考えよう.

```
?  T = x^4 + 24*x^2 + 585*x + 1791;
?  bnf = bnfinit(T);
?  bnf.sign
%68 [0,4]
?  bnf.clgp
%69 [4, [4], [[7, 4, 5, 6; 0, 1, 0, 0; 0, 0, 1, 0; 0, 0,
0, 1]]]
```

`bnf.no`=類数 $= 4$, `bnf.cyc`=位数 4 の巡回群，`bnf.gen`=各列がイデアルの基底を表す

一般の代数体でのイデアル

イデアルをどう表現するか：

- 考えている代数体 bnf の整数底 bnf.zk に関する $\mathbf{Z}$ 基底を HNF であらわしたもの（既出）．主に用いられる.
- idealprimedec() による表示.
- 代数体の整数環上 2 元で生成されるという性質を使う表示.

```
?  bnfisprincipal(bnf,idealpow(bnf,bnf.gen[1],4))
%190 = [[0]~, [3, -6, 2, -1]~]
```

bnf において，その類群の生成元 bnf.gen[1] を 4=bnf.no 乗したものが，単項か否かチェックした.

代数体の合成

polcompositum(pol1, pol2, flag=0: pol1 と pol2 とが定義する代数体の合成体を与える多項式のベクトルを与える.

```
?  polcompositum(x^2+23, polsubcyclo(9,3));
%82 = [x^6 + 63*x^4 + 2*x^3 + 1596*x^2 - 144*x + 1554]
?  qfb_23_3=bnfinit(%82[1]);
```

$\mathbf{Q}(\sqrt{-23})$ と，9 分体内の 3 次の部分体との合成体を定義している（$\mathbf{Q}(\sqrt{-23})$ の円分 $\mathbf{Z}_3$ 拡大の 1st layer）.

Q 上の Galois 拡大に関する計算

```
?  G17 = galoisinit(polcyclo(17));
?  galoisisabelian(G17)
%203 = [16]
?  galoispermtopol(G17,G17.gen[1])
%204 = x^7
?  galoisfixedfield(G17,G17.gen[1]^4,1)
%207 = x^4 + x^3 - 6*x^2 - x + 1
?  galoisexport(galoisinit(%))
%208 = "Group((1, 3, 2, 4))"
```

`polcyclo(17)` は円の 17 分体の多項式．その Galois 群は、位数 16 の巡回群．生成元 σ の根への作用は、7 乗．σ^4 の固定する体は、`x^4 + x^3 - 6*x^2 - x + 1` で定義される体．その体の $\mathbf{Q}$ 上の Galois 群を、GAP の記法で書くと、`Group((1, 3, 2, 4))` のようになる．

インクリメンタルな開発

コマンドラインで、対話的に開発を進めていく…….

gp の文法

条件分岐：`if()`

`if`(判定条件, 真の場合に実行する部分, 偽の場合に実行する部分)

くりかえし：`for()`

`for`(変数の初期化, 上限, 実行する文)

例：

```
?  for(d=1,100,if(isfundamental(d), h=qfbclassno(d);
if(Mod(h,3)==Mod(0,3),print(d, ", ", h))))
```

関数

```
関数名 (引数,...) =
{
        関数本体の定義
}
```

file 入出力

ファイルの読み込み \r file 名もしくは read(file 名)

ファイルへの書き出し write(file 名，...) ... の部分を計算し，その結果を file 名で指定されたファイルに書き出す

その他できること

- 代数体関連：ray class group/ray class field の計算．`bnr` 一族を使う
- 相対代数体の計算：`rnf` 一族を使う
- その他、楕円曲線（有理数体上、有限体上）.
- gp2C: gp のスクリプトを C に変換してくれる.
  - コンパイルして、ダイナミックリンクライブラリが作れる.
  - gp に組み込んで使う（単独の C プログラムにも組み込める）.
  - Unix 環境でしか使えない（？）

御清聴ありがとうございました

木村　巌