Introduction to PARI/GP (2/3)

The number fields module

Karim Belabas

http://pari.math.u-bordeaux.fr/

Notations and structures

Let $K = \mathbb{Q}[X]/(T)$ a number field of degree N, $\mathcal{O}_K$ its ring of integers ; $T \in \mathbb{Z}[X]$ is monic.

Three number field structures can be associated to K in GP :

$$\texttt{nf} < \texttt{bnf} < \texttt{bnr}$$

the last one being associated to K *and* a divisor $\mathfrak{f} = \mathfrak{f}_0\mathfrak{f}_\infty$. Obtained by `nfinit`, `bnfinit`, `bnrinit` applied to T, `nf` and $(\texttt{bnf}, \mathfrak{f})$ respectively. It is allowed to compute directly `bnfinit(T)`.

- `nf` contains basic invariants : maximal order given by $\mathbb{Z}$-basis `nf.zk`, absolute discriminant `nf.disc`, signature (r_1, r_2) `nf.sign`, defining polynomial `nf.pol`, its roots `nf.roots`
- `bnf` contains an `nf`, its ideal class group and unit group $\mathrm{Cl}(K)$ (`bnf.clgp`), $U(K)$ (`bnf.tu`, `bnf.fu`). Both are abelian groups given by generators and their order, and technical data required to solve the two associated discrete logarithm problems.
- `bnr` contains a `bnf`, a conductor $\mathfrak{f}$, the ray class group $\mathrm{Cl}_{\mathfrak{f}}(K)$, and data associated to the discrete logarithm problem in $\mathrm{Cl}_{\mathfrak{f}}(K)$.

Warnings (1/2)

- The only non-trivial task when computing an `nf` is to factor the discriminant. Shortcut :

$$\mathtt{nf} = \mathtt{nfinit}([\mathtt{T}, \mathtt{nfbasis}(\mathtt{T}, 1)])$$

  But then, the result is a priori incorrect if a prime larger than `primelimit` divides the *field* discriminant.

- `T = polredabs(T, 16)` will usually find a much nicer polynomial defining the same field K. (The 16 means : don't try to completely factor $\mathrm{disc}(T)$.) The result is a *canonical* polynomial defining K. An isomorphism test between $\mathbb{Q}[X]/(T)$ and $\mathbb{Q}[X]/(S)$ may be implemented as

$$\mathtt{polredabs}(\mathtt{S}) == \mathtt{polredabs}(\mathtt{T}),$$

  possibly with the 16 flag.

Warnings (2/2)

- Computing a `bnf` is a non-trivial randomized algorithm (may be *slow*). It uses a heuristic strengthening of the GRH. (For which there are scores of counter-examples, but which is usually correct.) Use

$$\texttt{bnf} = \texttt{bnfinit}(\mathtt{T},,[0.3, 12])$$

  to assume no more than the GRH (12 is the important constant here, improved bounds can be derived but this one is simple and universal). `bnfcertify(bnf)` removes the GRH assumption but is not practical for high degree fields.

- Computing a `bnr` is simple, *provided* the prime divisors of $\mathfrak{f}$ are small.

Elements in K

Assume that K is represented by one of the previous structures, associated to polynomial T. An element of K is given as

- a `t_INT`, `t_FRAC` or `t_POL` (implicitly modulo T), or
- a `t_POLMOD` (modulo T), or
- a `t_COL` `v` of dimension N, representing

  $$\texttt{sum(i = 1, N, v[i] * nf.zk[i])}$$

  so given in terms of the computed integral basis.

Missing : compact representation as $\prod x_i^{e_i}$ for some $x_i \in \mathcal{O}_K$, $e_i \in \mathbb{Z}$. Used intensively in PARI, but all GP routines returning an `nf` element return it in the last (`t_COL`) format.

(Fractional) Ideals in K (1/2)

- an element defines a principal ideal.
- the functions `idealprimedec` and `idealfactor` output maximal ideals in a special structure (`pr`) containing `pr.p`, `pr.e`, `pr.f`, `pr.gen`.
- a `t_MAT`, preferably in Hermite Normal Form (HNF, echelon form), represents the $\mathbb{Z}$-module generated by its columns (which represent `t_COL` elements of K). If more than N generators are provided, routines *assume* this is actually an $\mathcal{O}_K$-module

All formats are allowed as input, but most routines output ideals as HNF matrices. Finding a 2-element generating set $I = a\mathcal{O}_K + b\mathcal{O}_K$ is easy : `idealtwoelt(nf, I)`. Finding a single generator when I is principal is also easy, *provided* a `bnf` for K could be computed : `bnfisprincipal(bnf, I)`. This provides a factorisation $I = (\alpha) \prod g_i^{e_i}$, where the g_i are the generators of $\mathrm{Cl}(K)$ `bnf.gen` and $\alpha \in K$.

Divisors and congruence subgroups

A divisor $\mathfrak{f}$ is input as $[\mathfrak{f}_0, \texttt{arch}]$, where $\mathfrak{f}_0$ is an ideal and `arch` is a $\{0, 1\}$-vector with r_1 components, giving the coefficients of the real places (associated to the real roots of T, in the same order as `nf.roo`).

Hint : `bnrinit(bnf, f)` requires a `bnf` argument containing the fundamental units. Use `bnf = bnfinit(T, 1)` to make sure this is the case.

Hint : use `bnrinit(bnf, f, 1)` to include generators for $\mathrm{Cl}_{\mathfrak{f}}(K)$ (required by some class field theoretic routines).

`rnfkummer(bnr, , p)` returns defining polynomials for all class fields of (prime) degree p over K and conductor $\mathfrak{f}$.

`v = subgrouplist(bnr, [p], 1)` outputs the list of congruence subgroups of index p in $\mathrm{Cl}_{\mathfrak{f}}(K)$, and arbitrary conductor.

`v0 = subgrouplist(bnr, [p], 0)`, same with conductor $\mathfrak{f}$.

```
L = eval(setminus(Set(v), Set(v0)))
```

```
vector(#L, i, bnrconductor(bnr,L[i]));
```

```
vector(#L, i, rnfkummer(bnr,L[i]));
```