Number field sieve

Loïc Grenié

January 16^{th} 2013

1 The number field sieve

1.1 Introduction

We are trying to compute the class group of a number K field using Buchmann algorithm. Let n be the dimension of K, $\mathcal{O}_K$ its ring of integers. Basically one computes a limit T such that

$$\mathcal{B}_T = \{\mathfrak{p} : \mathfrak{p} \text{ is a prime ideal and } \operatorname{N}\mathfrak{p} \leqslant T\}$$

contains a set of generators of the class group of K and then a set E_T of elements of K divisible only by the primes in $\mathcal{B}_T$ and such that the ideal class group of K is isomorphic to

$$\langle \mathcal{B}_T \rangle / \langle (x) : x \in E_T \rangle \ .$$

The elements of $\mathcal{B}_T$ are called generators and the elements of E_T relations. If $\#E_T > \#\mathcal{B}_T$ products of elements of E_T yield units. Using analytic class number formula one can test whether the class group and the unit group of K have been computed.

The two main problems of the algorithm are finding relations (elements of E_T) and a linear algebra HNF problem yielding the class group and units through their logarithmic embeddings.

You can find more information in last year's `bnfinit()` seminar in *Atelier PARI/GP 2012*.

The number field sieve aim is to efficiently find elements of E_T.

1.2 Biasse and Fieker's sieve

Building on the quadratic sieve, Biasse and Fieker have suggested a way to find relations. Their basic algorithm, called **line sieving** is the following.

Init

1. Choose two positive integers I and J.

2. Choose two elements α and β of $\mathcal{O}_K$ (precisely the two first elements of an LLL basis of $\mathcal{O}_K$ for a random norm). The idea is to look search for elements of E_T of the form $x\alpha + y\beta$ with $(x, y) \in [-I, I] \times [0, J]$.
3. Initialize a table L of reals with an approximation of $\log |\operatorname{N}_{K/\mathbf{Q}}(x\alpha+y\beta)|$, for $(x, y) \in [-I, I] \times [0, J]$.
4. Let $y_0 \in [1, J]$. Then $P(x) = \operatorname{N}(x\alpha + y_0\beta)$ is a polynomial of degree (at most) n.

Sieve

5. Fix a prime p below one of the prime ideals of $\mathcal{B}_T$.
6. Let x_0 be one of the roots of P mod p. Then for all $k \in \mathbf{Z}$, $\operatorname{N}((x_0 + kp)\alpha + y_0\beta)$ is congruent to 0 mod p. Remove $\log p$ from $L[x_0 + kp, y_0]$ for all suitable k.
7. Iterate over x_0, p and y_0.

Relation recollection

8. Try to factor $x\alpha + y\beta$ for all (x, y) such that $L[x, y] \leqslant l_0$ for some limit l_0.

They also mention

- **Lattice sieving** in which the iteration over k is combined with iteration over y_0 through the use of a reduced basis of the lattice spanned by $(x_0, 1)$ and $(p, 0)$ in $[-I, I] \times [0, J]$
- **Special-q strategy**, where the main loop is over the "large" primes (above I).

We will have a false negative if $x\alpha + y\beta \in E_T$ but $L[x, y] > l_0$ after the loops and a false positive if $x\alpha + y\beta \notin E_T$ but $L[x, y] \leqslant l_0$. Note that both can happen if the initialization of $L[x, y]$ is too far from $\log \operatorname{N}_{K/\mathbf{Q}}(x\alpha + y\beta)$.

2 Our implementation

2.1 Some remarks

The algorithm is based on the fact that, for $t \in K$, $|\operatorname{N}_{K/\mathbf{Q}}(t)| = \prod_{\mathfrak{p}} \operatorname{N}\mathfrak{p}^{v_{\mathfrak{p}}(t)}$. In Step 6 we thus remove $\log p$ from the log of the norm of $x\alpha+y\beta$ when we have verified that p divides such norm. The limit l_0 accounts for the facts that we do not really factor $x\alpha + y\beta$ but find **some** of its prime factors in $\mathcal{B}_T$ and that we have an approximation of the logarithm of its absolute norm.

We see here a three possible additional imprecisions: when p divides the norm of t we know that some ideal $\mathfrak{p} \mid p$ divides t but

- it is possible that $\mathfrak{p} \notin \mathcal{B}_T$ if K is not Galois over $\mathbf{Q}$, case in which we should not remove $\log p$ from $L[x, y]$;
- the norm of $\mathfrak{p}$ could be greater than p or

- there could be more than one ideal above p in $\mathcal{B}_T$ dividing t. In the last two cases we should remove a multiple of $\log p$ from $L[x, y]$.

It would be much better to know which elements of $\mathcal{B}_T$ divide $x\alpha + y\beta$ and what is the log of their norm.

2.2 Norm

The log of the norm of each ideal of $\mathcal{B}_T$ can easily be computed beforehand, during the computation of $\mathcal{B}_T$. Indeed for each prime $p \leqslant T$, PARI computes $\log p$ and the inertial index of each $\mathfrak{p} \mid p$.

2.3 Divisibility

It is not very difficult to check whether a given element $t \in K$ is an divisible by a prime ideal $\mathfrak{p}$. We suppose that we have a $\mathbf{Z}$-basis $\mathcal{V}$ of $\mathcal{O}_K$, that we have a $\mathbf{Z}$-basis $\mathcal{H}$ of $\mathfrak{p}$ such that its coordinates in $\mathcal{V}$ are a matrix H in HNF form, that $\mathfrak{p} \cap \mathbf{Z} = p\mathbf{Z}$ and $\mathrm{N}\,\mathfrak{p} = p^f$.

There is a subset $\underline{f} = \{1, ...\}$ of $\{1, \dots, n\}$ such that $\#\underline{f} = f$ and if $H = (h_{ij})$ then the submatrix of H made of the rows and columns with indices in $\underline{f}$ is equal to pI_f and

$$\forall i, \quad i \notin \underline{f} \Rightarrow h_{ii} = 1 \ .$$

Then $\mathcal{H}$ is a $\mathbf{Q}$-basis of K and $t \in \mathfrak{p}$ if and if only its coordinates in this basis are integral. If t is identified with the column of its coordinates in $\mathcal{V}$, then its coordinates in $\mathcal{H}$ are $H^{-1}t$. Given the form of H explained above, to test the integrality of $H^{-1}t$ it is sufficient to check that the f coordinates with indices in $\underline{f}$ are integral.

This in turn can be done the following way. Let $C = C_{\mathfrak{p}}$ be the submatrix of pH^{-1} made of the rows with indices in $\underline{f}$. Then the coordinates of $H^{-1}t$ are integral if and only if the coefficients of Ct are multiple of p i.e.

$$t \in \ker C \mod p \ .$$

While we compute $\mathcal{B}_T$ we thus compute for each ideal $\mathfrak{p}$ the congruence matrix $C_{\mathfrak{p}}$.

Turning back to the sieving algorithm, suppose we have chosen α and β, let M be the $n \times 2$ matrix made of the coordinates of α and β in $\mathcal{V}$ and denote $\mathcal{Z} = \mathbf{Z}\alpha + \mathbf{Z}\beta \subset \mathcal{O}_K$. We then have $x\alpha + y\beta \in \mathfrak{p}$ if and only if (identifying elements of K with their coordinates in $\mathcal{V}$)

$$x\alpha + y\beta \in \ker C_{\mathfrak{p}} \mod p$$

which in turn is equivalent to

$$\begin{pmatrix} x \\ y \end{pmatrix} \in \ker C_{\mathfrak{p}} M \mod p \ .$$

At that point we are in a better situation because we can easily compute a $\mathbf{Z}$-basis

$$\begin{pmatrix} a & b \\ 0 & c \end{pmatrix}$$

in HNF form of the pullback of the lattice $\mathfrak{p} \cap \mathcal{Z}$. Moreover $\{a, c\} \subseteq \{1, p\}$ and $c = p \Rightarrow b = 0$ (and if $a = c = p$ then elements of $\mathfrak{p}$ are not interesting for our purpose because they are rational multiples of elements of $\mathcal{O}_K$). Informal benchmarks seem to indicate that computing $\ker C_{\mathfrak{p}} M$ for all $\mathfrak{p} \mid p$ is marginally faster than computing the roots modulo p of $\mathrm{N}(x\alpha + \beta)$ but this is not the main point of this observation: the point is that we get the precise ideal(s) that divide given $x\alpha + y\beta$ and the log of their norm.

2.4 Higher power divisibility

The same idea as above can easily be used for a prime-power ideal $\mathfrak{p}^r$. For each prime ideal $\mathfrak{p} \in \mathcal{B}_T$ compute $r = \left\lfloor \frac{\log 2IJ}{\log \mathrm{N}\,\mathfrak{p}} \right\rfloor$. Let H_r be the matrix of a basis of $\mathfrak{p}^r$ and $C_{\mathfrak{p},r}$ be the submatrix of $p^r H_r^{-1}$ consisting of the non-zero-mod-p^r rows. It is not difficult to compute a $\mathbf{Z}$-basis

$$\begin{pmatrix} a_r & b_r \\ 0 & c_r \end{pmatrix}$$

in HNF form of $\ker C_{\mathfrak{p},r} M$ i.e. of the pullback of the lattice $\mathfrak{p}^r \cap \mathcal{Z}$. If we are lucky enough that either $(a_r, c_r) = (p^{r-1}a, c)$ or $(a_r, c_r) = (a, p^{r-1}c)$, then for $1 \leqslant k \leqslant r$, we have $x\alpha + y\beta \in \mathfrak{p}^k$ if and only if (x, y) lies in the lattice generated by $\begin{pmatrix} a_r & b_r \\ 0 & c_r \end{pmatrix}$ mod p^k. If we are in one of these two lucky cases, easy congruences give the elements of all $\mathfrak{p}^k$. Otherwise we do not try to be more clever and just be happy with the elements of $\mathfrak{p}$.

The computation is fairly efficient. We get nearly exact valuation for all primes in $\mathcal{B}_T$ of all elements in $[-I, I]\alpha + [1, J]\beta$ in a few milliseconds.

Shortcomings: it is not very efficient for higher degrees ($n \geqslant 7$) or too small discriminant.

2.5 General organization

The general algorithm is as Function `algsieve` shown below. The logarithm of archimedean embeddings is described in the next paragraph.

Input: K, T, $\mathcal{B}_T$, $\{C_\mathfrak{p}\}$, $\{C_{\mathfrak{p},r}\}$, $S \subset \mathcal{B}_T$
Output: Some elements of K factorizable over $\mathcal{B}_T$

```
R ← ∅;
I ← ⌊T log log |Δ_K|⌋;
J ← ⌊log |Δ_K|⌋;
l_0 ← (1/2) log T;
N ← random_norm;
for 𝓘 ∈ {𝒪_K} ∪ S do
    B ← LLL(𝓘, N);
    α ← B[1];
    β ← B[2];
    if α is factorizable over 𝓑_T then
        R ← R ∪ {α};
    end
    M ← (α|β);
    L ← logarch(K, α, β, I, J);
    for 𝔭 ∈ 𝓑_T do
        [a, b, 0, c] ← ker C_𝔭 M   mod p;
        [a_r, b_r, 0, c_r] ← ker C_{𝔭,r} M   mod p^r;
        for j = c to J step c do
            for i = −I + ((I + bj)%a) to I step a do
                if Lucky case and i ≡ j a_r (mod p^2) then
                    L[i, j] ← L[i, j] − min(r, v_p(i − j a_r)) log N 𝔭;
                else
                    L[i, j] ← L[i, j] − log N 𝔭;
                end
            end
        end
    end
    for −I ⩽ i ⩽ I do
        for 1 ⩽ j ⩽ J do
            if L[i, j] ⩽ l_0 then
                if iα + jβ is factorizable over 𝓑_T then
                    R ← R ∪ {iα + jβ};
                end
            end
        end
    end
end
return R;
```

Function algsieve(K,T,$\mathcal{B}_T$,$\{C_\mathfrak{p}\}$,$\{C_{\mathfrak{p},r}\}$,$S \subset \mathcal{B}_T$)

2.6 Archimedean embedding

We used the following method to compute all $L[x,y] \simeq \log |\mathrm{N}\, K_{\mathbf{Q}}(x\alpha + y\beta)|$. First, observe that

$$L[x,y] = n \log y + \log \left|P\big(\frac{x}{y}\big)\right| \quad \text{where} \quad P(x) = \mathrm{N}_{K/\mathbf{Q}}(x\alpha + \beta)\ .$$

To compute $f(t) = \log |P(t)|$ we observe that

$$f'(t) = \frac{P'(t)}{P(t)}$$

and thus that f' changes sign where exactly one of P or P' changes sign.

We compute their square-free factorization of $Q_1 = \frac{P}{\gcd(P,P')}$ and $Q_2 = \frac{P'}{\gcd(P,P')}$. We thus have

$$P = \gcd(P, P') \prod_{i=1}^{k} Q_i^{v_i}$$
$$P' = \gcd(P, P') \prod_{i=k+1}^{k+l} Q_i^{v_i}$$

We then compute the real zeros of the Q_i's such that v_i is odd, using Uspensky method. These are the points $\{t_i\}$ where f' changes sign.

As a further optimization, observe that, if $\frac{\beta}{\alpha}$ is of degree n, then P changes sign at the real archimedean embeddings of $-\frac{\beta}{\alpha}$ thus we can save the computation of the real roots of P. If instead $\frac{\beta}{\alpha}$ is of degree lower than n then P is a power and Uspensky method is way faster.

We substitute the t_i with their best approximations from below and from above with rational numbers of denominators at most J and add $-I$ and I to the list. Then $f(t)$ is monotonous on each $[t_i, t_{i+1}]$. On the segment $[t_i, t_{i+1}]$ we compute $f(t)$ by dichotomy: we compute an approximation of $f(t)$ by linear interpolation if $|f(t_i) - f(t_{i+1})| \leqslant 2$ and cut the segment in half otherwise. As we can expect, we compute a lot of approximations of f near the roots of P.

The result is excellent: the error on the computation of the log of the norm is lower, usually much lower, than 2.

The slowest part is the computation of the real roots of P' and can become a significant part of the whole sieve if n is above 4. To dilute the problem we cannot take I and J too small.

The corresponding algorithm is given as Function `logarch` below.

Input: K, α, β, I and J

Output: A table L such that for $-I \leqslant x \leqslant I$ and $1 \leqslant y \leqslant J$,
$L[x, y] \simeq \log |\mathrm{N}_{K/\mathbf{Q}}(x\alpha + y\beta)|$

$L \leftarrow \texttt{array}(I, J)$;
$P \leftarrow \mathrm{N}(x\alpha + \beta)$;
$P1 \leftarrow P'$;
$D \leftarrow (P, P1)$;
$P \leftarrow P/D$;
$P1 \leftarrow P1/D$;
$T \leftarrow \texttt{concat}(\texttt{realroots}(\text{SQFF of P}), \texttt{realroots}(\text{SQFF of P1}))$;
$T \leftarrow$ `bestapprs(T,J)`;
$T \leftarrow \texttt{concat}([[-I], T, [I]])$;
$A \leftarrow \texttt{array}(\#T)$;
$A[1] \leftarrow \log |P(T[1])|$;
for $2 \leqslant i \leqslant \#T$ **do**
    $A[i] \leftarrow \log |P(T[i])|$;
    while $|A[i] - A[i-1]| > 2$ **do**
        Increase the size of T and A;
        $T[(i+1)..\#T] = T[i..(\#T-1)]$;
        $T[i] \leftarrow \frac{T[i-1]+T[i]}{2}$;
        $A[i] \leftarrow \log |P(T[i])|$;
    end
end
for $1 \leqslant j \leqslant J$ **do**
    $[k, r, dr] \leftarrow [0, 0, 0]$;
    for $-I \leqslant i \leqslant I$ **do**
        $r \leftarrow r + dr$;
        while $k < \#T$ *and* $\frac{i}{j} \geqslant T[k+1]$ **do**
            $k \leftarrow k + 1$;
            $dr \leftarrow \frac{A[k+1]-A[k]}{j(T[k+1]-T[k])}$;
            $r \leftarrow n \log j + A[k] + (i - j \cdot A[k])dr$;
        end
        $L[i, j] \leftarrow r$;
    end
end
return L

Function logarch(K, α, β, I, J)