A Multiple Polynomial General Number Field Sieve

Marije Elkenbracht-Huizing

CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract. We assume that the reader is familiar with the General Number Field Sieve (GNFS). This article describes a way to use more than two polynomials. Two, three and four polynomials are compared both for classical and for a special form of lattice sieving (line sieving). We present theoretical expectations and experimental results. With our present polynomial search algorithm, using more than two polynomials speeds up classical sieving considerably but not line sieving. Line sieving for two polynomials is the fastest way of sieving we tried so far.

1 Introduction

We assume that the reader is familiar with the General Number Field Sieve (GNFS [7] [3] [9]). We describe experiments following ideas of P.L. Montgomery in order to speed up the sieving stage of the GNFS algorithm by using more than two polynomials.

First, a short description of the GNFS is given and some notation is introduced. Secondly, the algorithm and the implementation of the multiple polynomial version of the GNFS is described, with special emphasis on the selection of the polynomials. Finally theoretical expectations are deduced and compared with experimental results.

2 Short Description of the GNFS

Let n be the number to be factored. We assume it is positive, odd and not a prime (power), which can easily be checked [10, §2.5]. In the GNFS two polynomials

$$f_i(x) = c_{i,d_i}x^{d_i} + c_{i,d_i-1}x^{d_i-1} + \cdots + c_{i,0} \in \mathbb{Z}[x] \quad (i = 1, 2)$$

are selected, having an integer m as common root modulo n. Both are irreducible over $\mathbb{Z}$, with $\mathrm{cont}(f_i) = \gcd(c_{i,d_i}, \ldots, c_{i,0}) = 1$ and $f_1 \neq \pm f_2$.

Let α_i be a root of $f_i(x)$ in $\mathbb{C}$ $(i = 1, 2)$ and let $\mathbb{Q}_n$ denote the ring of rational numbers with denominator coprime to n. Denote by $\phi_i : \mathbb{Q}_n[\alpha_i] \to \mathbb{Z}/n\mathbb{Z}$ $(i = 1, 2)$ two natural ring homomorphisms, determined by $\phi_i(\alpha_i) = m \bmod n$. Using the GNFS we will find a nonempty set $\mathcal{S} \subseteq \{(a, b) \in \mathbb{Z}^2 \mid a \text{ and } b \text{ coprime}\}$ such that both $\prod_{\mathcal{S}}(a - b\alpha_1)$ and $\prod_{\mathcal{S}}(a - b\alpha_2)$ are squares — β^2 and γ^2, say — in $\mathbb{Q}_n[\alpha_1]$ and $\mathbb{Q}_n[\alpha_2]$, respectively. Application of the homomorphisms on β^2 and γ^2 gives $\phi_1(\beta^2) \equiv \phi_2(\gamma^2) \bmod n$. This yields $(\phi_1(\beta))^2 \equiv (\phi_2(\gamma))^2 \bmod n$. When

$\phi_1(\beta)$ and $\phi_2(\gamma)$ are relatively prime to n, calculating $\gcd(n, \phi_1(\beta) - \phi_2(\gamma))$ will yield a non-trivial factor of n in at least half the cases.

After the first stage of the GNFS in which the polynomials are selected, several stages can be distinguished during which the set $\mathcal{S}$ is constructed. Denote by $F_i(x,y) = y^{d_i} f_i(x/y) \in \mathbb{Z}[x,y]$ the homogeneous form of $f_i(x)$ $(i = 1,2)$.

In the second stage, the *sieving*, the algorithm searches for $(a,b) \in \mathbb{Z}^2$ with a and b coprime, such that both integers $F_i(a,b)$ factor completely over the prime numbers below some user-determined *factor base bound* B_i, except for at most two primes, which should be between B_i and a *large prime bound* L_i. We call such integers $F_i(a,b)$ *smooth* and such (a,b)-pairs *relations*. By using lattice sieving [14], [7, §6] one of both integers $F_i(a,b)$ is allowed to have three primes between B_i and L_i. For a relation (a_j, b_j) we can write

$$F_1(a_j,b_j) = \pm \prod_{p \in \mathcal{K}_1} p^{e_1(j,p)} \qquad \text{and} \qquad F_2(a_j,b_j) = \pm \prod_{p \in \mathcal{K}_2} p^{e_2(j,p)}$$

where $e_i(j,p) \in \mathbb{N}$ and where $\mathcal{K}_1$ and $\mathcal{K}_2$ contain the prime numbers below L_1 and L_2, respectively.

To find the set $\mathcal{S}$ it is necessary to look at 'what kind of p' divides $F_i(a,b)$. For each prime number p we define the set

$$\mathcal{R}_i(p) = \{(r_1{:}r_2) \in \mathbf{P}^1(\mathbb{F}_p) \mid F_i(r_1,r_2) \equiv 0 \bmod p\},$$

where $\mathbf{P}^1(\mathbb{F}_p)$ denotes the projective line over $\mathbb{F}_p$. For a and b coprime, the integer $F_i(a,b)$ is divisible by a prime number p if and only if $(a \bmod p : b \bmod p) \in \mathcal{R}_i(p)$. Therefore the set $\mathcal{R}_i(p)$ partitions all (a,b)-pairs for which p divides $F_i(a,b)$ according to $(a \bmod p : b \bmod p)$. With

$$\mathcal{F}_i(C) = \{(p,(r_1{:}r_2)) \mid p \text{ prime},\ p < C,\ (r_1{:}r_2) \in \mathcal{R}_i(p)\},$$

for $C \in \mathbb{N}$, we can write for a relation (a_j, b_j)

$$F_1(a_j,b_j) = \pm \prod_{(p,(r_1:r_2)) \in \mathcal{F}_1(L_1)} p^{e_1(j,p,r_1,r_2)} \qquad \text{and}$$

$$F_2(a_j,b_j) = \pm \prod_{(p,(r_1:r_2)) \in \mathcal{F}_2(L_2)} p^{e_2(j,p,r_1,r_2)}$$

where $e_i(j,p,r_1,r_2) = e_i(j,p)$ if $(a_j \bmod p : b_j \bmod p) = (r_1 : r_2)$ and 0 otherwise. In order for $\prod_{\mathcal{S}}(a - b\alpha_i)$ to be a square in $\mathbb{Q}_n[\alpha_i]$, every exponent $\sum_{\mathcal{S}} e_i(j,p,r_1,r_2)$ in

$$\prod_{(a_j,b_j) \in \mathcal{S}} F_i(a_j,b_j) = \pm \prod_{(p,(r_1:r_2)) \in \mathcal{F}_i(L_i)} p^{\sum_{\mathcal{S}} e_i(j,p,r_1,r_2)} \tag{1}$$

should be even $(i = 1,2)$ [4, §12.7].

One can see from (1) that if we have a relation in which one of the integers $F_i(a,b)$ is divisible by a prime p to an odd power for a certain root $(a \bmod p : b \bmod p) = (r_1 : r_2) \in \mathcal{R}_i(p)$, and if this $(p,(r_1:r_2)) \in \mathcal{F}_i(L_i)$ is not occurring

in any other relation to an odd power, then this relation is useless and can be thrown away. After the sieving stage has gathered enough relations, the *filtering* stage performs this kind of data reduction. Details of this stage can be found in [7, §7].

In the next stage a matrix is built, which contains a column for every relation (a, b) (which survives the filtering stage) and a row for every element in $\{\mathcal{F}_1(L_1) \cup \mathcal{F}_2(L_2)\}$. An element of the matrix in the column of relation (a, b) and in the row of $(p, (r_1{:}r_2)) \in \mathcal{F}_i(L_i)$ is 1 when $(p|F_i(a, b)$ to an odd power and $(a \bmod p : b \bmod p) = (r_1 : r_2))$, and is 0 otherwise. Finding a vector in the nullspace of this matrix over $\mathbb{F}_2$ guarantees that, for the subset $\mathcal{T}$ of relations formed by the relations which correspond to a 1 in the solution vector, every exponent in (1) is even. By adding some extra rows coming from *quadratic characters* [1] [4, §8, §12.7], one is practically certain that the subset $\mathcal{T}$ is the wanted set $\mathcal{S}$. This *linear algebra* stage can be performed with the block Lanczos algorithm [12].

In the final stage of the GNFS algorithm we will know $\prod_{\mathcal{S}}(a - b\alpha_1) = \beta^2$ and $\prod_{\mathcal{S}}(a - b\alpha_2) = \gamma^2$, but we need β and γ to apply ϕ_i to. Therefore we have to extract square roots in $\mathbb{Q}_n[\alpha_i]$. For this *square root* stage we refer to [11] and [7, §9].

3 Algorithm of the Multiple Polynomial GNFS

The GNFS algorithm can be generalized to using more than two polynomials. Suppose we have k polynomials $f_1(x), f_2(x), \ldots, f_k(x) \in \mathbb{Z}[x]$ with an integer m as common root modulo n. Furthermore the polynomials should be irreducible over $\mathbb{Z}$, having $\mathrm{cont}(f_i) = 1$ $(i = 1, \ldots, k)$ and $f_i \neq \pm f_j$ for $(i \neq j)$.

We call an (a, b)-pair a (j_1, j_2)-relation $(1 \leq j_1 < j_2 \leq k)$ — and denote this by $(a, b)_{j_1,j_2}$ — if both integers $F_{j_1}(a, b)$ and $F_{j_2}(a, b)$ are smooth. For an (a, b)-pair with $t \geq 2$ smooth integers $F_{j_1}(a, b), F_{j_2}(a, b), \ldots, F_{j_t}(a, b)$ $(1 \leq j_1 < j_2 < \ldots < j_t \leq k)$ we can make $t - 1$ relations $(a, b)_{j_1,j_2}, (a, b)_{j_2,j_3}, \ldots, (a, b)_{j_{t-1},j_t}$. After the sieving in which we find the relations, we form a set $\mathcal{S}$ which is a subset of all relations such that

$$\frac{\prod_{(a,b)_{i,j_2} \in \mathcal{S}}(a - b\alpha_i)}{\prod_{(a,b)_{j_1,i} \in \mathcal{S}}(a - b\alpha_i)} = \beta_i^2 \tag{2}$$

where $\beta_i \in \mathbb{Q}_n[\alpha_i]$. On the one hand we have

$$\prod_i \phi_i \left(\frac{\prod_{(a,b)_{i,j_2} \in \mathcal{S}}(a - b\alpha_i)}{\prod_{(a,b)_{j_1,i} \in \mathcal{S}}(a - b\alpha_i)} \right) = \prod_i \phi_i(\beta_i^2) = \left(\prod_i \phi_i(\beta_i) \right)^2 \bmod n,$$

on the other hand we have

$$\prod_i \phi_i \left(\frac{\prod_{(a,b)_{i,j_2} \in \mathcal{S}}(a - b\alpha_i)}{\prod_{(a,b)_{j_1,i} \in \mathcal{S}}(a - b\alpha_i)} \right) = \frac{\prod_{(a,b)_{j_1,j_2} \in \mathcal{S}}(a - b\phi_{j_1}(\alpha_{j_1}))}{\prod_{(a,b)_{j_1,j_2} \in \mathcal{S}}(a - b\phi_{j_2}(\alpha_{j_2}))} = 1.$$

Calculating the gcd of n and $\prod_i \phi_i(\beta_i) - 1$ will split n into two non-trivial factors in at least half of the cases.

4 Implementation of the Multiple Polynomial GNFS

We based our implementation on the GNFS implementation described extensively in [7]. In this section we will describe the adjustments we made to let the implementation work for the multiple polynomial version of the GNFS.

During the sieving stage we use a *sieving interval* which corresponds to a range of a-values and one fixed value of b for which we want to check the smoothness of the integers $F_i(a,b)$. After the relations amongst these (a,b)-pairs have been selected, the sieving interval will correspond with a new range of a-values and/or a new value of b. In the sieving program we distinguish two cases, namely using two polynomials and using three or four polynomials. In the first case the sieving interval is represented by an array of unsigned shorts (2 bytes) and in the second case it is represented by an array of unsigned longs (4 bytes). Each block of bytes corresponds with one a-value. In a block the first byte is used to sum the logarithms of primes dividing $F_1(a,b)$, the second byte to sum the logarithms of primes dividing $F_2(a,b)$ and so on.

As usual in the first sieve procedure we add for each polynomial for all elements $(p,(r_1:r_2)) \in \mathcal{F}_i(B_i)$ the logarithms of the primes p for those a-values for which $p|F_i(a,b)$. We add the logarithms for polynomial f_i to the bytes corresponding with F_i. Afterwards we check for which a-values at least two of the values of $F_i(a,b)$ are sufficiently smooth, i.e. where the accumulated logarithms are at least $c_i(a,b) := \log(F_i(a,b)/T \cdot L_i^2)$. (The user-chosen constant T should compensate for not sieving over small primes and prime powers.) Therefore we split the sieving interval recursively in subintervals such that for each polynomial f_i the values of $c_i(a,b)$ do not vary more than a prescribed amount on the interval. Then we construct a threshold being an unsigned short and an unsigned long in the two respective cases, containing the thresholds $c_i(a,b)$ in the i-th byte. With only few operations we check for every a-value if the accumulated logarithms exceed the threshold for at least two polynomials. For these a-values, — (a,b) is now called a *candidate relation* — we use its entry of the sieving interval to store a positive integer which refers to an entry in another array. If $k > 2$ we also store the indices of the polynomials which are smooth. For other a-values we make the entry in the sieving interval zero. As opposed to our approach in [7] and corresponding with an idea in [6], we now carry out the sieving procedure once again. Instead of adding logarithms, we store for each candidate relation (a,b) the primes dividing one of the potentially smooth integers $F_i(a,b)$ and exceeding some user-determined bound themselves in the other array. Afterwards we calculate the potentially smooth integers $F_i(a,b)$ and divide out the stored primes. After dividing out the small primes we check if the remainders consist of at most two primes between B_i and L_i. When this is the case for t polynomials we have found $t-1$ relations.

In *classical* sieving, the sieving interval corresponds to a range of consecutive a-values. We start sieving with $b = 1$ and augment b until we have found enough relations.

In *lattice* sieving [14], we only sieve over pairs (a,b) of which we know that one $F_i(a,b)$, say $F_j(a,b)$, is divisible by a special large prime between user-chosen

bounds $L^{(l)}$ and $L^{(u)}$. The advantage is that the remaining part of $F_j(a,b)$ is more likely to be smooth. On the other hand we will miss the relations for which no $F_i(a,b)$ $(1 \leq i \leq k)$ has a prime in the interval $[L^{(l)}, L^{(u)}]$. Also the matrix will be larger, since the relations have more large primes on average. For the implementation of the lattice sieve we use an extra feature implemented in the classical way of sieving. There we have a possibility of sieving over a sublattice of the (a,b)-pairs. We can choose an integral, non-singular matrix M and sieve over pairs (a,b) of the form:

$$\begin{pmatrix} a \\ b \end{pmatrix} = M \begin{pmatrix} x \\ y \end{pmatrix},$$

while the program sieves over x and y. This is done by substituting the expressions of a and b in terms of x and y in $F_i(a,b)$ resulting in new polynomials $G_i(x,y)$, which are now the polynomials whose values should be smooth. The roots of the polynomials F_i have to be adapted to the roots of the polynomials G_i. When a pair (x,y) is a (j_1,j_2)-relation, the corresponding pair (a,b), together with j_1, j_2 and the primes dividing $G_{j_1}(x,y)$ and $G_{j_2}(x,y)$ and exceeding a user-chosen printing bound, are stored.

The lattice sieve sieves for every prime $L^{(l)} \leq q \leq L^{(u)}$, for a fixed value of b over all roots $(r_1{:}r_2) \in \cup\{\mathcal{R}_i(q)\}$ with $r_2 \neq 0$. When sieving over a root $(r_1{:}r_2)$ of $\mathcal{R}_j(q)$ we sieve only over the a-values with $a \equiv br_1r_2^{-1} \bmod q$, thus guaranteeing that $F_j(a,b)$ is divisible by q. This is the same as using a matrix

$$M = \begin{pmatrix} q & r_1r_2^{-1} \bmod q \\ 0 & 1 \end{pmatrix}$$

with y fixed to b and x in an interval such that $qx + b(r_1r_2^{-1} \bmod q)$ just fits in the a-interval. (Note that, when we compare our notation with that used in [14], we have $V_1 = (q,0), V_2 = (r_1r_2^{-1} \bmod q, 1)$ and that we are applying the 'sieving by rows' strategy.) $G_j(x,y)/q$ should be smooth over the primes below B_j, except for at most two large primes between B_j and L_j. The other $G_i(x,y)$ should be smooth over the primes below B_i, except for at most two large primes between B_i and L_i. Allowing primes equal to or larger than q to divide one of the $G_i(x,y)$ causes duplicate relations, but prevents missing a (i,j) or (j,i)-relation having two large primes smaller than q for $G_j(x,y)$ and a prime larger than q for $G_i(x,y)$. After we have sieved over all roots in $\cup\{(r_1{:}r_2) \in \mathcal{R}_i(q) | r_2 \neq 0\}$ we take the next value of b, and after we have sieved over all values of b we take the next prime in the interval $[L^{(l)}, L^{(u)}]$. We have implemented lattice sieving only for the case of quadratic polynomials. In §5 and [7] we explain how we select polynomials such that we can increase the efficiency of the sieving by taking a huge a-interval and $b = 1$. Therefore we call it *line* sieving.

The filtering stage is essentially the same as for two polynomials [7, §7]. We reduce the amount of data by checking which elements $(p,(r_1{:}r_2)) \in \mathcal{F}_i(L_i)$ occur in the collection of relations. For example, if an element $(p,(r_1:r_2)) \in \mathcal{F}_i(L_i)$ occurs at least once, then we verify in how many relations it occurs and to which power. As already said, if it occurs only once to an odd power, the relation is

useless and we throw it away. The other data reduction in the filtering stage is also done per set $\mathcal{F}_i(L_i)$.

In the linear algebra stage the matrix will have a column for every relation (a,b) and a row for each element in $\{\mathcal{F}_1(L_1) \cup \ldots \cup \mathcal{F}_k(L_k)\}$. An element of the matrix in the column of relation $(a,b)_{j_1,j_2}$ and in the row of $(p,(r_1{:}r_2)) \in \mathcal{F}_i(L_i)$ is 1 when $(i = j_1$ or $i = j_2$, $p|F_i(a,b)$ and $(a \bmod p : b \bmod p) = (r_1{:}r_2))$ and 0 otherwise. Add some character rows and let $\mathcal{S}$ be the subset of relations which corresponds to the 1's in the vector of the nullspace of the resulting matrix. Let $\mathcal{S}_i$ be the subset of $\mathcal{S}$ containing all $(a,b)_{j_1,j_2}$ relations for which $j_1 = i$ or $j_2 = i$. Then

$$\prod_{(a,b)\in\mathcal{S}_i} (a - b\alpha_i) = \gamma_i^2$$

with $\gamma_i \in \mathbb{Q}_n[\alpha_i]$, for $i = 1,\ldots,k$. We split $\mathcal{S}_i$ in two subsets: $\mathcal{S}_i^{(1)}$ containing all $(a,b)_{i,j_2} \in \mathcal{S}_i$ and $\mathcal{S}_i^{(2)}$ containing all $(a,b)_{j_1,i} \in \mathcal{S}_i$. Obviously

$$\prod_{(a,b)\in\mathcal{S}_i^{(1)}\cup\mathcal{S}_i^{(2)}} (a - b\alpha_i) = \gamma_i^2.$$

Hence

$$\frac{\prod_{(a,b)\in\mathcal{S}_i^{(1)}}(a - b\alpha_i)}{\prod_{(a,b)\in\mathcal{S}_i^{(2)}}(a - b\alpha_i)} = \beta_i^2 \tag{3}$$

which leads to (2).

In the square root stage, we extract the square root of $\beta_1^2, \beta_2^2, \ldots$, successively. Therefore the only change we have to make compared to applying GNFS with two polynomials, is not starting with $\prod_{\mathcal{S}}(a - b\alpha_i)$, but with (3). Actually, the square root stage sometimes replaces $(a,b)_{j_1,j_2}$ relations where $j_1 < j_2$ by $(a,b)_{j_2,j_1}$. This leads to more cancellation from the numerator and denominator in (3), an improvement even when using two polynomials.

5 Choice of the Polynomials

The polynomial selection is an important part of the multiple polynomial version of the GNFS. The polynomials $f_1, f_2, \ldots, f_k$ should be selected in such a way that the maximal values of $F_i(a,b)$ over all a and b-values in the sieving region will be as small as possible, to make them more likely to be smooth. When we use one linear polynomial f_1 and one higher degree polynomial f_2 as described in [3], [4, §12.2], the maximal values of $F_1(a,b)$ and $F_2(a,b)$ are unbalanced: the integer $F_2(a,b)$ will be much larger and much less likely to be smooth than $F_1(a,b)$ for most (a,b)-pairs. We therefore prefer to apply the multiple polynomial version of the GNFS by taking a set of equal-degree polynomials. When $d > 2$, it is an open problem to quickly construct two suitable polynomials of degree d and coefficients $O(n^{1/2d})$.

When applying Montgomery's method for selecting two quadratic polynomials one will find two polynomials for which $F_1(a,b)$ and $F_2(a,b)$ have the same order of magnitude. He observed that $f_1(x) = c_{1,2}x^2 + c_{1,1}x + c_{1,0}$ and $f_2(x) = c_{2,2}x^2 + c_{2,1}x + c_{2,0} \in \mathbb{Z}[x]$ have a common root m modulo n if and only if the vectors $\boldsymbol{a} = (c_{1,0}, c_{1,1}, c_{1,2})^T$ and $\boldsymbol{b} = (c_{2,0}, c_{2,1}, c_{2,2})^T$ are orthogonal to $(1, m, m^2)^T$ over $\mathbb{Z}/n\mathbb{Z}$ using the standard inner product. Suppose $f_1(x)$ and $f_2(x)$ are irreducible over $\mathbb{Z}$, cont(f_i)=1 ($i = 1, 2$) and that $f_1 \neq \pm f_2$. As will be explained further on, we can find in practice $\boldsymbol{a}$ and $\boldsymbol{b}$ of which the coefficients are appoximately $\mathcal{O}(n^{1/4})$, so the space orthogonal to $\boldsymbol{a}$ and $\boldsymbol{b}$ has rank 1 (both over $\mathbb{Z}$ and over $\mathbb{Z}/n\mathbb{Z}$). If $\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b}$ (cross product), then $\boldsymbol{c}$ must be a multiple of $(1, m, m^2)^T$ over $\mathbb{Z}/n\mathbb{Z}$. The fact that $f_1(x)$ and $f_2(x)$ are not multiples of each other ensures that $\boldsymbol{c}$ is not the zero vector. If $\boldsymbol{c} = (c_0, c_1, c_2)^T$, then c_0, c_1, c_2 is a geometric progression in $\mathbb{Z}/n\mathbb{Z}$. It is not a geometric progression over $\mathbb{Z}$, since then $f_1(x)$ and $f_2(x)$ would have a common factor $(x - m)$ over $\mathbb{Z}$.

Montgomery's algorithm for finding $f_1(x)$ and $f_2(x)$ reverses this construction and starts with a vector $\boldsymbol{c} = (c_0, c_1, c_2)^T \in \mathbb{Z}^3$, where c_0, c_1, c_2 is a geometric progression with ratio m over $\mathbb{Z}/n\mathbb{Z}$, but not over $\mathbb{Z}$. The vector $\boldsymbol{c}$ can be constructed as follows: for p prime such that $p < \sqrt{n}$ and n a quadratic residue modulo p, choose c_1 such that $c_1^2 \equiv n \bmod p$ and $|c_1 - n^{1/2}| \leq p/2$. The elements of $\boldsymbol{c} = (p, c_1, (c_1^2 - n)/p)^T$ form a geometric progression with ratio c_1/p over $\mathbb{Z}/n\mathbb{Z}$, not over $\mathbb{Z}$. Furthermore $c_i = \mathcal{O}(n^{1/2})$ (i=0,1,2). Take $s \in \mathbb{Z}/p\mathbb{Z}$ such that $c_1 s \equiv 1 \bmod p$. With $c_2 = (c_1^2 - n)/p$, the vectors

$$\boldsymbol{a'} = \begin{pmatrix} c_1 \\ -p \\ 0 \end{pmatrix} \text{ and } \quad \boldsymbol{b'} = \begin{pmatrix} (c_1(c_2 s \bmod p) - c_2)/p \\ -(c_2 s \bmod p) \\ 1 \end{pmatrix}$$

are both orthogonal to $\boldsymbol{c}$. From $\boldsymbol{a'} \times \boldsymbol{b'} = -\boldsymbol{c}$ and $\gcd(c_0, c_1, c_2) = 1$ we deduce that $\boldsymbol{a'}$ and $\boldsymbol{b'}$ span the sublattice of $\mathbb{Z}^3$ orthogonal to $\boldsymbol{c}$. Denote by $(\boldsymbol{a}, \boldsymbol{b})$ the inner product of $\boldsymbol{a}$ and $\boldsymbol{b}$ and remember that $\frac{(\boldsymbol{a},\boldsymbol{b})}{(\boldsymbol{a},\boldsymbol{a})}\boldsymbol{a}$ is the projection of $\boldsymbol{b}$ on $\boldsymbol{a}$. By reducing the basis $\{\boldsymbol{a'}, \boldsymbol{b'}\}$, one can find 'small' vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ with $\{\boldsymbol{a}, \boldsymbol{b}\}$ a basis of the sublattice of $\mathbb{Z}^3$ orthogonal to $\boldsymbol{c}$, such that $\left|\frac{(\boldsymbol{a},\boldsymbol{b})}{(\boldsymbol{a},\boldsymbol{a})}\right| \leq \frac{1}{2}$ and $\left|\frac{(\boldsymbol{a},\boldsymbol{b})}{(\boldsymbol{b},\boldsymbol{b})}\right| \leq \frac{1}{2}$. The angle θ between these vectors will be between 60 and 120 degrees. Since the surface of the parallelogram spanned by $\boldsymbol{a}$ and $\boldsymbol{b}$ is both equal to $\| \boldsymbol{a} \times \boldsymbol{b} \|$ and $\| \boldsymbol{a} \| \cdot \| \boldsymbol{b} \| \sin\theta$, we have

$$\| \boldsymbol{a} \| \cdot \| \boldsymbol{b} \| = \frac{\| \boldsymbol{c} \|}{\sin\theta} \leq \frac{2 \| \boldsymbol{c} \|}{\sqrt{3}} = \mathcal{O}(\| \boldsymbol{c} \|) = \mathcal{O}(n^{1/2}).$$

In practice both $\| \boldsymbol{a} \|$ and $\| \boldsymbol{b} \|$ are $\mathcal{O}(n^{1/4})$. For different values of p we will get a different pair of polynomials.

When using line sieving (explained in Section 4) we like to use a large range of a-values, say $|a| < M$ and only $b = 1$. For stimulating $F_i(a, 1) = c_{i,2}a^2 + c_{i,1}a + c_{i,0}$ to be smooth over the prime numbers below B_i, we would prefer

$c_{i,2} = \mathcal{O}(n^{1/4}/M)$, $c_{i,1} = \mathcal{O}(n^{1/4})$ and $c_{i,0} = \mathcal{O}(n^{1/4}M)$ rather than all of them being $\mathcal{O}(n^{1/4})$. How this is achieved is described in [7, §5].

For each value of p we construct a set of k polynomials as follows. In order to find more than two suitable polynomials we start with a pair of polynomials f_1 and f_2 as above. We look at linear combinations with small integer coefficients of these polynomials like $f_1, f_2, f_1 \pm f_2, 2f_1 \pm f_2, 3f_1 \pm f_2$. All these polynomials have m as common root modulo n, and their coefficients all have the same order of magnitude. Each linear combination is rated for having many roots modulo small primes and with the integral of $\log|F_i(x,y)|$, where (x,y) runs through the sieving region for (a,b), small. As can be read in [7, §5] one also likes the polynomials to have two real roots. The k linear combinations with the best rating are selected, and their individual ratings are summed to get the rating of the set. We choose the set with the best overall rating, over all considered primes p.

6 Free Relations

The Galois group of an irreducible polynomial $f_i(x) \in \mathbb{Z}[x]$ of degree 2 is the symmetric group S_2. For approximately $1/|S_2| = 1/2$ of the primes $q < L_i$, the polynomial $F_i(x,y)$ splits into two linear factors modulo q [5, §2, Theorem 1], [13, p. 566]. If such a prime q does not divide the discriminant of $f_i(x)$, then $f_i(x)$ splits into two *different* linear factors modulo q. If $t \geq 2$ polynomials $F_{j_1}(x,y)$, $\ldots, F_{j_t}(x,y)$ $(1 \leq j_1 < \ldots < j_t \leq k)$ split into two different linear factors modulo q and if q does not divide their leading coefficients, we call q a *free* prime and we have found $t-1$ free relations $p_{j_1,j_2}, \ldots, p_{j_{t-1},j_t}$. This terminology comes from the fact that we can select the ones which are smaller than $\min(B_i)$ without extra effort when calculating the factor bases $\mathcal{F}_i(B_i)$ and the ones larger than $\min(B_i)$ during the filtering stage [7, §7]. They are said to give rise to free relations because we now require

$$\frac{\left(\prod_{p_{i,j_2} \in \mathcal{T}} p\right)\left(\prod_{(a,b)_{i,j_2} \in \mathcal{S}} (a - b\alpha_i)\right)}{\left(\prod_{p_{j_1,i} \in \mathcal{T}} p\right)\left(\prod_{(a,b)_{j_1,i} \in \mathcal{S}} (a - b\alpha_i)\right)}$$

to be a square in $\mathbb{Q}_n[\alpha_i]$, where $\mathcal{T}$ is a suitably chosen subset of the set of free relations. It can be verified that the argumentation of §3 still holds. How these free relations should be treated in the stages after the sieving is described in [7]. When the extension fields of the polynomials are algebraically independent, $L = L_1 = \ldots = L_k$ and

$$t(k) = \left(\frac{1}{2}\right)^k \sum_{l=2}^{k} \binom{k}{l}(l-1) = \frac{k}{2} - 1 + 2^{-k}, \tag{4}$$

then there are $t(k)\pi(L)$ free relations, where $\pi(L)$ is the number of primes below L.

7 Theoretical Expectations and Experimental Results

7.1 Classical Sieving

Theoretical expectations

As described in Section 5, the $F_i(a,b)$'s of the k polynomials are chosen to have the same degree and same order of magnitude. Therefore it is natural to make all factor base bounds B_i and large prime bounds L_i equal, to B and L say. $|\mathcal{F}_i(C)|$ is approximately the number $\pi(C)$ of primes below C [8, Chapter VIII,§4], so all $|\mathcal{F}_i(B_i)|$ are approximately equal $(i = 1, \ldots k)$, which also holds for all $|\mathcal{F}_i(L_i)|$. We will discuss here the advantages and the disadvantages of using more than two polynomials, compared with two polynomials:

1. In the sieve program most time is spent in the two sieve procedures. The first is the part in which for each polynomial and for all elements $(p, (r_1 : r_2)) \in \mathcal{F}_i(B)$ we add the logarithms of the primes p for those a-values where $p|F_i(a,b)$. In the second the primes themselves are stored instead of the sum of their logarithms. When using $k > 2$ polynomials instead of two polynomials, we expect that the time spent in these procedures will grow with a factor $k/2$ per point (a,b) sieved.
2. As will be explained further on, when using $k > 2$ instead of two polynomials, we will find more candidate relations. For each candidate relation we spend some time with dividing stored primes and small primes from the hopefully smooth integers $F_i(a,b)$ and checking if the remainders consist of at most two primes between B and L. The time spent in this part of the program will increase. However we expect it to remain small in comparison with the time spent in both sieve procedures.
3. Another disadvantage is that we need more relations when using $k > 2$ polynomials. We need at least $(|\mathcal{F}_1(L)| + \ldots + |\mathcal{F}_k(L)| + |\text{character rows}|)$ relations to ensure that we find a vector in the null space of the matrix. The number of character rows is negligible compared to the number of other rows. When using $k > 2$ polynomials instead of two polynomials, the number of relations needed therefore grows with a factor $k/2$.
4. Furthermore, there are three practical disadvantages. First, the program needs to keep in memory the elements of all $\mathcal{F}_i(B)$. Using more memory can slow down the sieving process. Secondly, the use of 4 bytes per a-value may hurt cache performance. Thirdly, using more than two polynomials will lead to more relations and a larger matrix and therefore to increasing time needed for the filtering, the block Lanczos and the square root stages. Because we cannot handle at the moment unlimited large matrices with block Lanczos even extra relations and filtering effort can be necessary to reduce the matrix size.
5. The most important advantage of using $k > 2$ polynomials instead of two polynomials is that the number of expected relations per (a,b)-pair will grow with the number of polynomials. Let p be the probability that one of the $F_i(a,b)$ is smooth. Then the number of expected relations for k polynomials

is about $\binom{k}{2}p^2$. Hence the expected number of relations is increasing quadratically with k, while the needed number of relations and the time spent in both sieve procedures is only increasing linearly in k.

6. Another advantage is the increasing amount of free relations which is available when using more than two polynomials. According to 3. we need $k\pi(L)$ relations, but (4) implies $t(k)\pi(L)$ of them are free. Hence we need we need $(2-1/4)\pi(L)$, $(3-5/8)\pi(L)$ and $(4-17/16)\pi(L)$ non-free relations when using two, three and four polynomials, respectively. Therefore, when using four polynomials instead of two, we do not need twice as many, but only 68% more non-free relations.
7. A last advantage is that we can reduce the sieving region, leading to smaller values of $F_i(a,b)$. On the one hand the coefficients for the $k>2$ polynomials will increase by a factor $\sqrt{k}$, because we take linear combinations of two polynomials. On the other hand, when finding $\binom{k}{2}$ as many relations, but only needing $(k-t(k))/(2-t(2))$ as many, we can take the sieving region a factor $c(k)=1.75\binom{k}{2}/(k-t(k))$ smaller. Because relatively more relations are found in the area near the origin, this factor is pessimistic. Suppose that the maximal values of a and b both decrease by a factor $\sqrt{c(k)}$, then the maximal value of the integers $F_i(a,b)$ will decrease by a factor $c(k)/\sqrt{k}$.

Taking into account these advantages and disadvantages (apart from 2, 4 and 7 because they are partly machine dependent and difficult to quantify) we expect to speed up the sieving by a factor $1.75(k-1)/(k-t(k))$ (1.47 for $k=3$, 1.79 for $k=4$) when using $k>2$ instead of two quadratic polynomials.

Experimental results

For all experiments in this article we sieved for the 96-digit number $(80^{61}-1)/(79\cdot 37699\cdot 57250710187259)$ [2]. For the constant T — described in §4 — we took 500, and for the lower bound for the large primes in the factor base — which we store in the second sieving procedure — 700. For classical sieving we optimized the searching of polynomials for $|a|\le 2\cdot 10^7$ and $1\le b\le 10^6$ (2 polynomials), $1\le b\le 452,000$ (3 polynomials) and $1\le b\le 280,000$ (4 polynomials), thereby applying the factor $1.75\binom{k}{2}/(k-t(k))$ we found in the previous section. The polynomials are given in Table 5. The common root m can be found with the Maple command '$(\mathrm{Gcd}(f_1,f_2) \bmod n)-n$;'.

In the first experiment we sieved on a collection of 60 SGI machines at CWI. Most machines are SGI Indy workstation's (100 MHz R4000SC processor). We sieved with 2, 3 and 4 polynomials. For all three experiments we took the factor base bounds $B=2.4\cdot 10^6$ and a small large prime bound of $L=10^7$ in an attempt to keep the matrix small. To decrease the time needed for the experiments we sieved over 100 intervals of 100 consecutive b-values, which were equally divided over the b-interval. According to our experience this can be used to give a good estimate for the sieving time needed to sieve over all values of b. For calculating the number of needed relations we used the formula $(k-t(k))\pi(L)$ which we derived in the previous section. The value of $\pi(L)$ is approximated by $L/(\ln L-1)$. In Table 1 we give the results of the experiments. We see that with these bounds,

	2 polynomials	3 polynomials	4 polynomials
sieving region	$\|a\| \leq 2 \cdot 10^7$	$\|a\| \leq 2 \cdot 10^7$	$\|a\| \leq 2 \cdot 10^7$
	$1 \leq b \leq 10^6$	$1 \leq b \leq 452,000$	$1 \leq b \leq 280,000$
# b-values over which was sieved	10,000	10,000	10,000
# relations found	11,294	44,992	83,811
sieving time	1,080,333 sec.	1,993,971 sec.	2,636,647 sec.
estm. # rel. over all b's	1,129,400	2,033,638	2,346,708
relations needed	1,157,553	1,570,965	1,943,036
estm. total sieving time	30,757 hours	19,340 hours	16,980 hours
speedup factor		1.59	1.81
theoretical speedup factor		1.47	1.79

Table 1. Classical sieving with small L.

the number of extra relations found per b-value grows faster than with a factor $\binom{k}{2}$. First, because the sieving region is smaller, also the average value of $F_i(a,b)$ is smaller. Secondly, the occurrence of more than two smooth values of $F_i(a,b)$'s for an (a,b)-pair was not taken into account. Therefore we could have taken the b-interval even smaller than applying the factor $1.75\binom{k}{2}/(k-t(k))$. Optimizing the polynomial search program for a smaller b-interval, would have led to polynomials with a smaller average value of the $F_i(a,b)$ on that interval. This would have favored the speedup factors, which are already above the predicted ones.

Using a larger large prime bound speeds up the sieving, but leads to a larger matrix. However, the size of the matrix can be reduced by sieving more relations than strictly needed. To compare the previous experiment to sieving with a larger large prime bound, we took the factor base bounds $B = 1.2 \cdot 10^6$ and large prime bounds $L = 2*10^7$. We sieved over 100 intervals of 50 consecutive b-values, which were again equally divided over the b-interval. In Table 2 we give the results of the experiments. For all polynomials the estimated total sieving time is less than in the previous experiment. Again we could have reduced the sieving region for 3 and 4 polynomials, which would have led to better results. The speedup factor for using more than 2 polynomials is better than in the previous experiment. When using smaller factor base bounds the program uses less memory, what is especially in favor for the experiments for 3 and 4 polynomials.

7.2 Line Sieving

Theoretical expectations

Compared with classical sieving we notice two differences. Suppose we sieve over a root $(r_1{:}r_2)$ of $\mathcal{R}_j(q)$. Let p be the probability that $F_i(a,b)$ is smooth $(1 \leq i \leq k, i \neq j)$ and $p_q > p$ be the probability that $F_j(a,b)/q$ is smooth. The number of expected relations per root for k polynomials is about $(k-1)pp_q + \binom{k-1}{2}p^2$ which is smaller than $\binom{k}{2}pp_q$. Hence the expected number of relations is still increasing more than linearly, but less fast than classical sieving.

On the other hand there is also a new advantage. The larger the prime q is,

the larger p_q, the more relations we find and the less time we spend per relation. When using k polynomials, the number of roots of q grows with a factor $k/2$. When keeping $L^{(u)}$ equal, we can take $L^{(l)}$ larger and the average time needed per relation will decrease.

Experimental results
For line sieving we optimized the searching of polynomials for $|a \cdot q| \leq 2.4 \cdot 10^{13}$ and $b = 1$. The polynomials are given in Table 5. In the experiments we started sieving at $q = 10^7$ and decreased q until we had found enough relations. The a interval was adapted such that $|a \cdot q|$ was approximately $2.4 \cdot 10^{13}$ and $2 \cdot |a|$ was a multiple of $8 \cdot 10^5$, $9 \cdot 10^5$ or 10^6, which was taken as the sieving interval. The factorbase bound was $2,400,000$.

We sieved on 1 processor of an SGI Challenge (150 MHz R4400SC processors) for the interval $[L^{(l)}, L^{(u)}] = [9.99 \cdot 10^6, 10^7]$ for 2, 3 and 4 polynomials, and for the second and third best of the set of 4 polynomials. This interval contains 614 primes. We give the results in Table 3. From this experiment we conclude that the factor by which the number of relations per root increases (when we go from 2 to k polynomials) is smaller than the expected factor $k - 1$. From experiment 2b we can conclude that these polynomials were much worse than the polynomials from experiment 2a. That we still get a factor less than $k - 1$ can be explained by the varying results for the four polynomials of experiment 4. The number of norms which were smooth were 8,926, 8,601, 8,209 and 6,854, respectively. As already described we took the middle two for experiment 2b. Apparently, when searching for a good set of four polynomials for line sieving, the individual polynomials are much worse than for a good set of two. It is still an open question why this is only occurring for line sieving. Because the costs for sieving are still growing with a factor $k/2$, the time needed per relation is increasing. Since we even need more relations, we can conclude that in the case of line sieving with our present polynomial search algorithm, the use of three or four polynomials does not give better results than two polynomials.

To compare line sieving for two polynomials to classical sieving with four polynomials we sieved again at the collection of 60 SGI machines at CWI. We sieved the interval $[L^{(l)}, L^{(u)}] = [7 \cdot 10^6, 10^7]$, after which we had found enough relations. The results are stated in Table 4. Comparing the sieving times of Table 4 with Table 2, we conclude that line sieving is the fastest way of sieving we tried.

8 Conclusions

Using more than two polynomials in our experiments speeds up classical eving considerably, but with our present polynomial search algorithm it is not useful for line sieving. Line sieving with two polynomials turns out to be much faster than classical sieving with two, three or four polynomials.

Acknowledgements

The author is particularly grateful to P.L. Montgomery for his suggestions and for sharing his NFS program (partially developed by A.K. Lenstra and Oregon State University) with her. The author thanks A.K. Lenstra, P.L. Montgomery, H.J.J. te Riele and R. Tijdeman for reading the paper and for suggesting several improvements. This work was sponsored by the Stichting Nationale Computerfaciliteiten (National Computing Facilities Foundation, NCF) for the use of supercomputer facilities, with financial support from the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (Netherlands Organization for Scientific Research, NWO).

	2 polynomials	3 polynomials	4 polynomials
sieving region	$\lvert a\rvert \leq 2 \cdot 10^7$	$\lvert a\rvert \leq 2 \cdot 10^7$	$\lvert a\rvert \leq 2 \cdot 10^7$
	$1 \leq b \leq 10^6$	$1 \leq b \leq 452,000$	$1 \leq b \leq 280,000$
# b-values over which was sieved	5,000	5,000	5,000
# relations found	9,974	40,412	74,441
sieving time	445,037 sec.	794,651 sec.	1.064,276 sec.
estm. # rel. over all b's	1,994,800	3,653,245	4,168,696
relations needed	2,213,614	3,004,191	3,715,710
estm. total sieving time	27,436 hours	16,409 hours	14,756 hours
speedup factor		1.67	1.86
theoretical speedup factor		1.47	1.79

Table 2. Classical sieving with large L.

	2 polynomials	3 polynomials	4 polynomials	2 of 4 pol.
number of experiment	2a	3	4	2b
# roots for these primes	1,230	1,842	2,492	1,252
# relations found	4,131	10,056	16,297	2,792
# rel. per root	3.36	5.46	6.54	2.23
sieving time	8,903 sec.	23,986 sec.	44,290 sec.	8,909 sec.
sieving time per rel.	2.16 sec.	2.39 sec.	2.72 sec.	3.19 sec.

Table 3. Line sieving on small interval.

	2 polynomials
sieving region	$7 \cdot 10^6 \leq q \leq 10^7$
# relations found	1,255,429
# duplicates found	70,605
sieving time	1,492 hours
relations needed	1,157,553
matrix size	$727,551 \times 755,014$

Table 4. Line sieving.

Polynomials for classical sieving

Two polynomials:

$f_1 = -13120\,43224\,23539\,33603\,04\,x^2 - 36831\,71061\,38731\,26822\,0586\,x - 47944\,69584\,67939\,17526\,1751$

$f_2 = -12842\,68073\,51211\,20626\,9127\,x^2 - 37374\,96629\,28900\,17466\,00744\,x - 50908\,58246\,15164\,78021\,37304\,4$

Three polynomials:

$f_1 = -81457\,47262\,62603\,41067\,52\,x^2 - 21907\,43902\,78265\,15386\,61562\,x - 15722\,10378\,07940\,49403\,82024\,3$

$f_2 = 63100\,65717\,54229\,79090\,55\,x^2 - 65557\,80218\,25966\,18826\,2644\,x + 12421\,85964\,80634\,29350\,20870\,0$

$f_3 = 27075\,94441\,52529\,27833\,917\,x^2 + 22400\,98373\,04752\,97387\,3630\,x + 52987\,68272\,49843\,37454\,44634\,3$

Four polynomials:

$f_1 = 12390\,73187\,55161\,44127\,368\,x^2 + 24239\,10314\,43974\,74435\,446\,x + 63206\,80527\,88089\,61740\,74862\,5$

$f_2 = 25404\,30588\,60748\,46488\,41\,x^2 - 95312\,13792\,13776\,74856\,6134\,x + 13329\,24536\,09396\,75769\,56883\,9$

$f_3 = -47694\,40109\,69369\,01808\,45\,x^2 - 28836\,03240\,78572\,77201\,33848\,x - 23219\,06919\,59899\,34432\,04210\,8$

$f_4 = -17160\,17198\,52098\,34308\,213\,x^2 - 29078\,42343\,93012\,51945\,69294\,x - 86425\,87447\,47988\,96172\,79073\,3$

Polynomials for line sieving

Two polynomials:

$f_1 = -61752\,28709\,x^2 - 28439\,56167\,76472\,53488\,3012\,x - 23150\,75391\,96708\,44709\,46738\,43001\,01431\,04$

$f_2 = 36722\,19603\,49\,x^2 - 95804\,69208\,15565\,40216\,59734\,x + 55577\,38537\,72556\,82485\,08362\,87488\,04948\,557$

Three polynomials:

$f_1 = -61752\,28709\,x^2 - 41557\,09722\,49053\,21746\,2146\,x - 60322\,80472\,99241\,35463\,24848\,00868\,95834\,81$

$f_2 = 36722\,19603\,49\,x^2 - 17798\,72103\,80668\,80253\,65960\,x - 47522\,34791\,89843\,68635\,47624\,22338\,54755\,04$

$f_3 = 10594\,00435\,59\,x^2 - 17014\,79960\,59970\,45883\,54376\,x - 17670\,15952\,52792\,48411\,38213\,60976\,34047\,784$

Four polynomials:

$f_1 = -64231\,53716\,9\,x^2 - 32161\,96143\,83407\,36549\,48282\,x - 31858\,25118\,64795\,48549\,18090\,37721\,41973\,945$

$f_2 = 14026\,33092\,71\,x^2 - 31647\,58500\,09956\,15741\,82098\,x + 11159\,93053\,42414\,53761\,24237\,19053\,94304\,731$

$f_3 = 63795\,67362\,7\,x^2 - 10506\,33029\,72197\,78513\,30184\,x + 73048\,25321\,47390\,14462\,82730\,27492\,61248\,00$

$f_4 = -13107\,82555\,9\,x^2 - 32033\,36732\,90044\,56347\,56736\,x - 21103\,70575\,62992\,97971\,57508\,48527\,57904\,276$

Table 5. Polynomials used for classical and line sieving experiments.

References

1. L.M. Adleman. Factoring numbers using singular integers. In *Proceedings 23rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 64–71, New Orleans, 1991.
2. R.P Brent, P.L. Montgomery, and H.J.J. te Riele. Update 2 to: Factorizations of $a^n \pm 1, 13 \leq a < 100$. Technical Report NM-R9609, Centrum voor Wiskunde en Informatica, Amsterdam, 1996.
3. J. Buchmann, J. Loho, and J. Zayer. An implementation of the general number field sieve. In D.R. Stinson, editor, *Advances in Cryptology - CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 159–165, Berlin, 1994. Springer–Verlag.
4. J.P. Buhler, H.W. Lenstra, Jr., and C. Pomerance. Factoring integers with the number field sieve, pages 50–94 in [9].
5. F.G. Frobenius. Über Beziehungen zwischen den Primidealen eines algebraischen Körpers und den Substitutionen seiner Gruppe. *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin*, pages 689–703, 1896. Also in Ferdinand George Frobenius, Gesammelte Abhandlungen, Band II, Springer-Verlag, Berlin, 1968.
6. R.A. Golliver, A.K. Lenstra, and K.S. McCurley. Lattice sieving and trial division. In L.M. Adleman and M.-D. Huang, editors, *Algorithmic Number Theory*, volume 877 of *Lecture Notes in Computer Science*, pages 18–27, Berlin, 1994. Springer–Verlag.
7. R.M. Huizing. An implementation of the number field sieve. Technical Report NM-R9511, Centrum voor Wiskunde en Informatica, Amsterdam, 1995. To appear in *Experimental Mathematics*.
8. S. Lang. *Algebraic Number Theory*. Addison-Wesley, Reading, MA, USA, 1970.
9. A.K. Lenstra and H.W. Lenstra, Jr. *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer–Verlag, Berlin, 1993.
10. A.K. Lenstra, H.W. Lenstra, Jr., M.S. Manasse, and J.M. Pollard. The factorization of the ninth Fermat number. *Mathematics of Computation*, 61:319–349, 1993.
11. P. L. Montgomery. Square roots of products of algebraic numbers. In Walter Gautschi, editor, *Mathematics of Computation 1943–1993: a Half-Century of Computational Mathematics*, pages 567–571. Proceedings of Symposia in Applied Mathematics, American Mathematical Society, 1994. Long version to appear.
12. P. L. Montgomery. A block Lanczos algorithm for finding dependencies over GF(2). In L.C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology – EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 106–120, Berlin, 1995. Springer–Verlag.
13. J. Neukirch. *Algebraische Zahlentheorie*. Springer–Verlag, Berlin, 1992.
14. J.M. Pollard. The lattice sieve, pages 43–49 in [9].