

A Reduction Algorithm for Algebraic Function Fields

Mark van Hoeij, Andrew Novocin
Department of Mathematics
Florida State University
Tallahassee, FL 32306, USA.
e-mail: {hoeij,anovocin}@math.fsu.edu

April 8, 2008

Abstract

Computer algebra systems often produce large expressions involving complicated algebraic numbers. In this paper we study variations of the `polred` algorithm that can often be used to find better representations for algebraic numbers. The main new algorithm presented here is an algorithm that treats the same problem for the function field case.

1 Introduction

Consider the following problem: Given a finite extension K of \mathbb{Q} find a $\beta \in K$ with $\mathbb{Q}(\beta) = K$ for which the minimal polynomial m_β has “small” integer coefficients. Here “small” means not much larger than optimal.

A good and practical solution to this problem is the `polred` algorithm, by Cohen and Diaz y Diaz [1]. This algorithm involves integer factorization (for computing the ring of algebraic integers \mathcal{O}_K), floating point arithmetic (except in the totally real case) as well as the LLL ([8]) algorithm. The topics of this paper are the following

- Topic 1. Can integer factorization be avoided?
- Topic 2. Can floating point arithmetic be avoided in general?
- Topic 3. How to generalize `polred` to the function field case, where K is a finite extension of $\mathbb{Q}(x)$?

We will start by explaining the `polred` algorithm in section 2.1. Topics 1 and 2 and an example will be discussed in section 2.2. The remainder of the paper will be devoted to topic 3.

2 The number field case

2.1 The polred algorithm

Suppose K , a finite extension of \mathbb{Q} , has r_1 real embeddings $\sigma_1, \dots, \sigma_{r_1}$ and r_2 pairs of complex embeddings $\sigma_{r_1+1}, \overline{\sigma_{r_1+1}}, \dots, \sigma_{r_1+r_2}, \overline{\sigma_{r_1+r_2}}$. Combined there are $n := [K : \mathbb{Q}] = r_1 + 2r_2$ embeddings $K \rightarrow \mathbb{C}$. Then we have a \mathbb{Q} -linear map

$$(\sigma_1, \dots, \sigma_{r_1+r_2}) : K \rightarrow \mathbb{R}^{r_1} \oplus \mathbb{C}^{r_2}.$$

We can identify $\mathbb{R}^{r_1} \oplus \mathbb{C}^{r_2}$ with \mathbb{R}^n by splitting real and imaginary parts. Denote the resulting map as

$$\psi : K \rightarrow \mathbb{R}^n. \tag{1}$$

We can now define two bilinear forms $K \times K \rightarrow \mathbb{R}$ as follows $\langle a, b \rangle_1 := \text{Tr}(ab)$ and $\langle a, b \rangle_2 := \psi(a) \cdot \psi(b)$ where Tr denotes the trace over K/\mathbb{Q} and \cdot is the usual dot product in \mathbb{R}^n . Note that $\langle, \rangle_1 = \langle, \rangle_2$ if and only if K is totally real (i.e. $r_2 = 0$), if and only if the bilinear form \langle, \rangle_1 is positive definite.

The `polred` algorithm, by Cohen and Diaz y Diaz [1], works as follows. First, compute a basis b_1, \dots, b_n of \mathcal{O}_K as \mathbb{Z} -module. This way we can represent elements of \mathcal{O}_K by elements of \mathbb{Z}^n . Then compute floating point approximations for $\psi(b_1), \dots, \psi(b_n)$. These approximations give us a bilinear form close to \langle, \rangle_2 . Now compute an LLL reduced basis of \mathcal{O}_K with respect to this bilinear form. This way LLL finds elements in \mathcal{O}_K whose images under ψ are “small”, i.e. not much larger than optimal (LLL finds vectors that are no more than $2^{n/2}$ times longer than optimal). Now select the first $\beta \notin \mathbb{Q}$ found by LLL.

Remark 1 *A potential difficulty is that β generates a non-trivial subfield of K . This problem is easily avoided by taking a small linear combination of elements of the LLL basis. However, one might not want to actively avoid finding useful information (a non-trivial subfield). Instead of avoiding this situation, it would be better to make use of this subfield. However, for simplicity we will assume that β does generate K (using a small linear combination if necessary).*

Given that the image of $\beta \in \mathcal{O}_K$ under ψ is “small”, one concludes that the complex roots of the minimal polynomial m_β are small, after all, the real and imaginary parts of these roots are exactly the entries of $\psi(\beta)$. This in turn implies that the coefficients of m_β are “small” (meaning that they can be bounded in terms of the size of an optimal minimal polynomial). This algorithm is not guaranteed to find an optimal minimal polynomial (optimal in terms of bitsize or in terms of coefficient size) but like LLL it works very well in practice.

The `polred` algorithm is very useful for computer algebra systems in order to simplify expressions containing large complicated algebraic numbers. However, popular computer algebra systems such as Maple and Mathematica do not use these techniques. A possible explanation may be the use of integer factorization that is involved in the computation of \mathcal{O}_K . Factoring a large integer may take too long. However, in almost all examples this potential bottleneck can be easily avoided, see subsection 2.2 below. Topic 2 (avoiding floating point arithmetic) will be discussed as well in subsection 2.2 as well.

2.2 Topics 1 and 2, and an example

Consider the system of equations given in [6]. A computer algebra system produced the following solution (this solution can easily be found using standard Gröbner basis techniques since the system of equations in [6] was not complicated).

$$\begin{aligned}
 y &= \frac{582483877536562478229177673229}{15974159158723631332001909574} - \frac{4583573815816690393633870284023}{5814593933775401804848695084936} \alpha \\
 &+ \frac{119532100283469028894215401}{15974159158723631332001909574} \alpha^2 - \frac{211625248935690502697605}{532471971957454377333969858} \alpha^3 \\
 &+ \frac{939747984163203177097}{7987079579361815666000954787} \alpha^4 - \frac{2403470531291923165}{15974159158723631332001909574} \alpha^5, \\
 z &= \frac{\alpha}{728}, \\
 x &= -\frac{554100881976373365591535043891}{15974159158723631332001909574} + \frac{4360115737681067598970904726867}{5814593933775401804848695084936} \alpha \\
 &- \frac{111409542439749934294396505}{15974159158723631332001909574} \alpha^2 + \frac{193496497879991531164401}{532471971957454377333969858} \alpha^3 \\
 &- \frac{838972747013492420704}{7987079579361815666000954787} \alpha^4 + \frac{2077032365296943155}{15974159158723631332001909574} \alpha^5
 \end{aligned}$$

where α has minimal polynomial $m_\alpha(x) = x^6 - 967x^5 + 409231x^4 - 98821664x^3 + 14505901608x^2 - 1221461035503x + 45330846457297$.

The first problem is to avoid factoring integers, which is used in the algorithm for computing a basis of \mathcal{O}_K . Note that the `polred` algorithm could also be applied to a subring $R \subseteq \mathcal{O}_K$ instead of \mathcal{O}_K itself. Experiments show that the smaller the index of R in \mathcal{O}_K is (i.e. the larger R is), the better the result will be. In the example, the index of $\mathbb{Z}[\alpha]$ in \mathcal{O}_K is large; the ring $\mathbb{Z}[\alpha]$ is too small to find an generator of K with a nice minimal polynomial. The key idea in avoiding integer factorization is now the following: instead of using just the information stored in α and m_α , we should use the information stored in all of x, y, z . We first multiply $x, y, z, 1/x, 1/y, 1/z$ by rational numbers q_1, \dots, q_6 chosen so that the products $a_1, \dots, a_6 := q_1x, q_2y, \dots, q_6 \cdot 1/z$ are algebraic integers. So we have six elements $a_1, \dots, a_6 \in \mathcal{O}_K$. Then we compute $R = \mathbb{Z}[a_1, \dots, a_6]$ to obtain a subring of \mathcal{O}_K that is much larger than $\mathbb{Z}[\alpha]$. This R is found with no integer factorization.

For `polred` to work well we need to use either \mathcal{O}_K or a large subring of \mathcal{O}_K (i.e. a subring with small index in \mathcal{O}_K). The above construction for R usually leads to a large subring of \mathcal{O}_K provided that more than one *independently* constructed elements (x, y, z in the example) of K were given. In the example R is equal to \mathcal{O}_K .

By computing such ring R instead of \mathcal{O}_K we can use `polred` without having to factor integers, except in examples where only one element of K is given. However, given an expression (such as the solution of a system of equations)

that involves only one element of K , there is less motivation to simplify this expression than in situations where several elements of K occur.

If $r_2 = 0$ (all embeddings are real) then `polred` does not need any floating point computations because in this case the two bilinear forms \langle, \rangle_1 and \langle, \rangle_2 are the same (recall that the definition of \langle, \rangle_1 does not involve \mathbb{R}). If $r_2 > 0$ then \langle, \rangle_1 is no longer positive definite, and since the LLL algorithm works for positive definite bilinear forms, `polred` will apply LLL with the bilinear form \langle, \rangle_2 which involves computing in \mathbb{R} .

There exist variations of the LLL algorithm that do allow bilinear forms that are not positive definite, see [7] and [10]. Because of this, one can write a modification of `polred` that never needs floating point arithmetic. We implemented this modification, and found that it has an advantage as well as a disadvantage. The advantage is that it makes the algorithm faster. The disadvantage is that \langle, \rangle_1 is less closely related to the size of m_β as \langle, \rangle_2 . So one saves computation time, at the cost of finding a less good m_β . In the example we found $m_\beta(x) = x^6 - 2x^5 + 3x^4 - 7849x^3 + 85730x^2 - 481151x + 5038077$ which is less good than the m_β (given below) that one would otherwise find.

One possible strategy may be to first use the above variation of `polred` that avoids floating point arithmetic, obtaining a partial improvement, and then follow up on that by additional reduction using floating point arithmetic. This gives the same result as the standard `polred`. Unfortunately, it does so in almost the same amount of CPU time; in experiments there was little gain in this hybrid approach. We find $m_\beta(x) = x^6 - x + 1$.

Using this m_β , while keeping track of the relation between α and β during the computation of m_β , one can rewrite the expressions for x, y, z in terms of β and we arrive at the improved solution:

$$\begin{aligned} z &= \frac{107\beta^5}{728} + \frac{107\beta^4}{728} + \frac{69\beta^3}{728} + \frac{17\beta^2}{364} + \frac{9}{91}, \\ y &= \frac{165\beta^5}{728} + \frac{165\beta^4}{728} + \frac{103\beta^3}{728} - \frac{\beta^2}{364} + \frac{15}{182}, \\ x &= -\frac{209\beta^5}{728} - \frac{209\beta^4}{728} - \frac{179\beta^3}{728} - \frac{23\beta^2}{364} + \frac{163}{182} \end{aligned}$$

3 The function field case

We will now consider the function field case. Given a finite extension K of $\mathbb{Q}(x)$, we want to find a generator β of K whose minimal polynomial over $\mathbb{Q}(x)$ has “small” coefficients, both in terms of the degrees with respect to x , as well as in terms of the sizes of the coefficients in \mathbb{Q} .

Denote R_∞ as the ring of all $a \in \mathbb{Q}(x)$ that have no pole at $x = \infty$ (the set of rational functions a for which the numerator does not have higher degree than the denominator). Then $\mathbb{Q}[x] \cap R_\infty = \mathbb{Q}$ since the constants are the only functions $a \in \mathbb{Q}(x)$ with no poles in $\overline{\mathbb{Q}} \cup \{\infty\}$. A function $a \in \mathbb{Q}(x)$ has a pole at $x = \infty$ of order d if and only if d is the smallest integer for which $a \in x^d R_\infty$.

Let $f \in \mathbb{Q}[x, y]$ be irreducible and let $K = \mathbb{Q}(x)[y]/(f)$. Let $n = [K : \mathbb{Q}(x)] = \deg_y(f)$. We will assume that $n > 1$. We can write $K = \mathbb{Q}(x, \alpha)$ where α denotes $y \bmod f$, the image of y in K . Denote \mathcal{O}_K as the integral closure of $\mathbb{Q}[x]$ in K , and denote \mathcal{O}_∞ as the integral closure of R_∞ in K . One can compute a basis b_1, \dots, b_n of \mathcal{O}_K as a $\mathbb{Q}[x]$ -module. Available implementations for this are based on the round two or round four algorithm (see [3]) or on Puiseux expansions (see [4]). These implementations also allow to compute a basis b'_1, \dots, b'_n of \mathcal{O}_∞ as R_∞ -module. A convenient way to do this is to do a change of variable $x \mapsto 1/t$, compute a local integral basis at $t = 0$ (most implementations have this feature), and then to change back $t \mapsto 1/x$.

A place is a discrete valuation on K over \mathbb{Q} . Every non-singular point on the algebraic curve defined by f is a place, while a singular point corresponds to one or more (but finitely many) places. One can effectively compute with places by using Puiseux expansions. The ring of functions with no poles at the finite places (the places where x has no pole) is \mathcal{O}_K . The ring of functions with no poles at the infinite places (the places where x has a pole) is \mathcal{O}_∞ . So the intersection $\mathcal{O}_K \cap \mathcal{O}_\infty$ is the set of functions with no poles on the curve; the set of locally constant functions. The dimension of this set is the number of components, i.e., the number of irreducible factors of f in $\overline{\mathbb{Q}}[x, y]$, see [9]. We will assume that f remains irreducible over $\overline{\mathbb{Q}}$, in other words we assume that $\mathcal{O}_K \cap \mathcal{O}_\infty = \mathbb{Q}$.

Given f and K , the goal is now the following: Find some $\beta \in K$ that generates K over $\mathbb{Q}(x)$ whose minimal polynomial

$$m_\beta = Y^n + a_{n-1}Y^{n-1} + \dots + a_0 \in \mathbb{Q}(x)[Y] \quad (2)$$

has “small” coefficients. Here we will define “small” in the following way:

- Condition 1: The coefficients a_i in equation (2) are in $\mathbb{Q}[x]$.
- Condition 2: The total degree of m_β as a bivariate polynomial is n . In other words $\deg_x(a_i) \leq n - i$.
- Condition 3: If there is a β satisfying the previous assumptions for which the degree of m_β as a polynomial in x is less than n , then we should find such β . Moreover, the coefficients of the a_i in \mathbb{Q} should be “small”, meaning that their numerators and denominators are not much larger than optimal under the previous requirements.

In this section we will focus on the first two conditions, which can be reformulated as

1. $\beta \in \mathcal{O}_K$
2. $\beta \in x\mathcal{O}_\infty$

(to see the equivalence of condition 2, note that $\beta \in x\mathcal{O}_\infty$ iff $\beta/x \in \mathcal{O}_\infty$ iff $m_{\beta/x} = Y^n + \frac{a_{n-1}}{x}Y^{n-1} + \dots + \frac{a_0}{x^n} \in R_\infty[Y]$ iff $\deg_x(a_i) \leq n - i$ for all i).

There need not be any β that generates K over $\mathbb{Q}(x)$ and that satisfies these two conditions. For simplicity we will assume that such β does exist.

Remark 2 *If such β does not exist, this would not complicate matters much. We could simply replace the second condition by a weaker condition, namely require that $\deg_x(a_i) \leq 2(n-i)$ for all $i \in \{0, \dots, n-1\}$, or equivalently, that β is in $x^2\mathcal{O}_\infty$. If that fails as well, we could weaken the condition further to $\beta \in x^3\mathcal{O}_\infty$, etc.*

For conditions 1 and 2 we need $\beta \in \mathcal{O}_K \cap x\mathcal{O}_\infty$. For condition 3 see subsection 3.2. An algorithm to compute $\mathcal{O}_K \cap x\mathcal{O}_\infty$ for a special case was given in Section 3.2 in [5]. The next subsection presents an efficient method to compute $\mathcal{O}_K \cap x\mathcal{O}_\infty$ in general.

3.1 Normalize an integral basis at infinity

The process of normalizing an integral basis at infinity was introduced in [11] as one of the steps in the integration of algebraic functions. For completeness we will give a description of this process.

Algorithm: A basis of \mathcal{O}_K that is normal at infinity.

1. Let b_1, \dots, b_n be a basis of \mathcal{O}_K as $\mathbb{Q}[x]$ -module.
2. Let b'_1, \dots, b'_n be a basis of \mathcal{O}_∞ as R_∞ -module.
3. Write $b_i = \sum_{j=1}^n r_{ij}b'_j$ with $r_{ij} \in \mathbb{Q}(x)$.
4. Let $D \in \mathbb{Q}[x]$ be a non-zero polynomial for which $a_{ij} := Dr_{ij} \in \mathbb{Q}[x]$ for all i, j . Now $Db_i = \sum_{j=1}^n a_{ij}b'_j$.
5. For each $i \in \{1, \dots, n\}$, let m_i be the maximum of the degrees of $a_{i,1}, \dots, a_{i,n}$. Now let $V_i \in \mathbb{Q}^n$ be the vector whose j 'th entry is the x^{m_i} -coefficient of a_{ij} . Let $d_i := m_i - \deg_x(D)$.
6. If V_1, \dots, V_n are linearly independent, then return b_1, \dots, b_n and stop. Otherwise, take $c_1, \dots, c_n \in \mathbb{Q}$, not all 0, for which $c_1V_1 + \dots + c_nV_n = 0$.
7. Among those $i \in \{1, \dots, n\}$ for which $c_i \neq 0$, choose one for which d_i is maximal. For this i , do the following
 - (a) Replace b_i by $\sum_{k=1}^n c_k x^{d_i - d_k} b_k$.
 - (b) Replace a_{ij} by $\sum_{k=1}^n c_k x^{d_i - d_k} a_{kj}$ for all $j \in \{1, \dots, n\}$.
8. Go back to step 5.

The b_1, \dots, b_n remain a basis of \mathcal{O}_K throughout the algorithm because the new b_i in step 7a can be written as a nonzero rational number times the old b_i plus a $\mathbb{Q}[x]$ -linear combination of the b_j , $j \neq i$. When we go back to step 5 the non-negative integer d_i decreases while the d_j , $j \neq i$ stay the same. Hence the algorithm must terminate.

Let b_1, \dots, b_n be the output of the algorithm. By construction, the number d_i in the algorithm is the smallest integer for which $b_i \in x^{d_i}\mathcal{O}_\infty$. If $\beta \in \mathcal{O}_K$

with $\beta \neq 0$ then we can write $\beta = c_1 b_1 + \dots + c_n b_n$ for some $c_1, \dots, c_n \in \mathbb{Q}[x]$. Denote d_β as the maximum of $\deg_x(c_j) + d_j$ taken over all j for which $c_j \neq 0$. Then $\beta \in x^{d_\beta} \mathcal{O}_K$ by construction. Since the vectors V_1, \dots, V_n in the algorithm are linearly independent when the algorithm terminates, there can not be any cancelation, which means that d_β is the smallest integer for which $\beta \in x^{d_\beta} \mathcal{O}_K$. Because of this, we get the following:

Let b_1, \dots, b_n be the output of the above algorithm. If d is a positive integer, then the set $B_d := \{x^j b_i \mid 0 \leq j \leq d - d_i\}$ is a basis of $\mathcal{O}_K \cap x^d \mathcal{O}_K$ as \mathbb{Q} -vector space.

We may assume that B_1 contains the elements 1 and x since both are in $\mathcal{O}_K \cap x \mathcal{O}_K$. We will assume that B_1 contains a generator of K over $\mathbb{Q}(x)$, otherwise the first two conditions in the previous section can not be met. In particular this assumption implies that B_1 should have more than two elements (of course one could weaken condition 2 by taking B_d instead of B_1 where d is the smallest positive integer for which $\mathcal{O}_K \cap x^d \mathcal{O}_K$ contains a generator of K over $\mathbb{Q}(x)$, see also the remark at the end of the previous section).

3.2 Condition 3

Let V denote the vector space $\mathcal{O}_K \cap x \mathcal{O}_K$. The previous section described how to compute a basis B_1 of V . Assume that V contains a generator of K over $\mathbb{Q}(x)$. Experiments show that taking a random $\beta \in V$ (or taking some $\beta \in B_1$ that generate K) almost always leads to a minimal polynomial $m_\beta = Y^n + a_{n-1} Y^{n-1} + \dots + a_0$ whose bitsize is very much larger than optimal (the coefficients of the a_i are much larger than optimal).

This can be remedied in much the same way as in the `polred` algorithm. By evaluating functions at places P_1, \dots, P_s on the curve, we get a maps $\phi_i : V \rightarrow K_{P_i}$, where K_{P_i} is the residue field at P_i , which is an algebraic number field. Combining these maps we get a map $\sigma : V \rightarrow \bigoplus_{i=1}^s K_{P_i}$. If $N := [K_{P_1} : \mathbb{Q}] + \dots + [K_{P_s} : \mathbb{Q}]$ is large enough ($N \geq n$ certainly suffices) then σ must be one to one. By mapping each K_{P_i} to \mathbb{R}^{n_i} where $n_i = [K_{P_i} : \mathbb{Q}]$ as in equation (1) in section 2.1, we get a map $\Psi : V \rightarrow \mathbb{R}^N$.

Now let R_i be $\mathcal{O}_{K_{P_i}}$ or a large subring of $\mathcal{O}_{K_{P_i}}$ if we want to avoid factoring integers, like in section 2.2. The set of all $\beta \in V$ whose image in K_{P_i} ends up in R_i for all i is a lattice $L \subset V$ (i.e. a \mathbb{Z} -module in V of rank $\dim(V)$). The map Ψ induces a dot-product on L (and on V), coming from the dot-product on \mathbb{R}^N . This dot-product takes values in \mathbb{R} . One can now compute an LLL reduced basis of L with respect to this dot-product and select the first element (see also Remark 1) that generates K over $\mathbb{Q}(x)$.

The heuristic justification behind this approach is the following. We find with LLL some $\beta \in L \subset V$ whose evaluations are reasonably nice, i.e., not much worse than optimal (remember that LLL does not necessarily find the shortest vectors, but it does find vectors that are at most be some bounded factor larger than the shortest vectors). In addition, if there is some $\beta \in L$

that generates K over $\mathbb{Q}(x)$ for which m_β is nice, then the evaluations of β will also be reasonable nice. If we use enough places P_1, \dots, P_s then we would expect that only those $\beta \in L$ with nice m_β could correspond to short vectors in the lattice. This (by no means precise) argument makes it plausible that this approach leads to some $\beta \in V$ for which the expression m_β is not very much larger than optimal. Like `polred`, this approach gives very good results in practice.

The places we propose to use are the infinite places as well as the places above $x = 0$ (in the unlikely event that the map σ above is not one to one, we can add the places above $x = 1$, $x = -1$, etc.). For finite places P_i , one maps $\beta \in V$ to the value of β at P_i (i.e. the image of β in the residue field at P_i). The places P_i at infinity are treated slightly differently; to get a map $V \rightarrow K_{P_i}$ when P_i is a place at infinity, we need to send $\beta \in V$ to the value at P_i of β/x instead of β . This is because we allowed β (but not $\beta/x \in \mathcal{O}_\infty$) to have a poles at infinite places P_i .

So far we have not yet discussed the part of condition 3 that discusses the degree of m_β viewed as polynomial in x . This issue is handled as follows. Suppose P_i is a place at infinity, and consider the map $V \rightarrow K_{P_i}$ where β maps to the value of β/x at P_i . Let V_i be the kernel of this map. If $V_i \neq V$ but V_i still contains a generator of K , then we will replace V by V_i . We repeat this as long as we can still find a place at infinity for which “evaluation, and taking the kernel” leads to a new V of lower dimension that still contains a generator. Each time such a step is possible, the maximal possible $\deg_x(m_\beta)$ decreases by $n_i = [K_{P_i} : \mathbb{Q}]$.

We give an example where K is the extension of $\mathbb{Q}(x)$ given by the minimal polynomial

$$m_\alpha(y) = x^3y^5 - (16x - 5)x^3y^4 + 2(43x^2 - 37x + 13)x^3y^3 + 2(40x^5 - 180x^4 + 232x^3 - 135x^2 + 50x - 8)x^2y^2 + (-80x^9 - 160x^8 + 868x^7 - 1648x^6 + 1516x^5 - 834x^4 + 255x^3 - 30x^2 - 8x + 2)y + 32x^{11} - 592x^8 + 80x^9 - 2100x^6 + 1709x^5 + 1608x^7 - 845x^4 + 215x^3 - 4x^2 + 64x^{10} - 13x + 3.$$

In general, a generator of K better than the given one (namely α) need not exist, in which case our algorithm can not do much. In this example, however, the field K does have a much better generator over $\mathbb{Q}(x)$ than α , so the algorithm is useful here. In this example, it finds $\beta \in K$ with $m_\beta(y) = y^5 + 2xy + x^2$. A partial implementation, including this example and a few other examples, is available at <http://www.math.fsu.edu/~hoeij/files/NormalBasis/>

References

- [1] H. Cohen, *A Course in Computational Algebraic Number Theory*, Springer-Verlag, (1993).
- [2] H. Cohen and F. Diaz y Diaz, *A polynomial reduction algorithm*, Séminaire de Théorie des Nombres Bordeaux **3**, 351-360 (1991).

- [3] D. Ford and P. Letard, *Implementing the Round Four Maximal Order Algorithm*, Journal de Théorie des Nombres de Bordeaux **6**, 39-80 (1994).
- [4] M. van Hoeij, *An algorithm for computing an integral basis in an algebraic function field.*, J. Symb. Comput., **18**, 353-363 (1994).
- [5] M. van Hoeij, *Rational Parametrizations of Algebraic Curves using a Canonical Divisor.* J. Symb. Comput., **23**, 209-227 (1997).
- [6] M. van Hoeij, A. Novocin, *Equations for the Example*, <http://www.math.fsu.edu/~anovocin/ISSAC2005.txt> (2005).
- [7] G. Ivanyos, Á. Szántó, *Lattice basis reduction for indefinite forms and an application*, Discrete Mathematics **153**, 177-188 (1996).
- [8] Lenstra, Lenstra, Lovász, *Factoring Polynomials with Rational Coefficients*, Math. Ann. **261**, 515-534 (1982).
- [9] J.-F. Rago, *Sur la factorisation absolue des polynômes*. Université de Limoges, (1997).
- [10] Denis Simon, *Solving quadratic equations using reduced unimodular quadratic forms*, to appear in Math. Comp.
- [11] B. Trager, *Integration of algebraic functions*, Ph.D. thesis, Dept. of EECS, MIT, (1984).