S. Leem et al. *Res. Number Theory (2025)11:17*
https://doi.org/10.1007/s40993-024-00606-6

Research in Number Theory

RESEARCH

Solving norm equations in global function fields

Sumin Leem[1], Michael J. Jacobson Jr.[2] and Renate Scheidler[1*]

*Correspondence:
rscheidl@ucalgary.ca
[1]Department of Mathematics and Statistics, University of Calgary, 2500 University Drive NW, Calgary, AB T2N 1N4, Canada
[2]Department of Computer Science, University of Calgary, 2500 University Drive NW, Calgary, AB T2N 1N4, Canada
Full list of author information is available at the end of the article

Abstract

We present two new algorithms for solving norm equations over global function fields with at least one infinite place of degree one. The first of these is a substantial improvement of a method due to Gaál and Pohst, while the second approach uses index calculus techniques and is significantly faster asymptotically and in practice. Both algorithms incorporate compact representations of field elements which results in a significant gain in performance compared to the Gaál–Pohst approach. We provide Magma implementations, analyze the complexity of all three algorithms under varying asymptotics on the field parameters, and provide empirical data on their performance.

Keywords: Global function field, Norm equation, S-unit, Index calculus

Mathematics Subject Classification: Primary 11Y16, Secondary 11Y40, 11R58, 11D57, 11G20

1 Introduction

Solving norm equations, which are a special instance of Diophantine equations over global fields, is a classical research area in number theory. In the setting of number fields, this problem has undergone extensive investigation. In 1973, Siegel proposed a method for solving norm equations in Galois fields by bounding the absolute values of the solutions [25]. In 1989, Pohst and Zassenhaus introduced a technique for solving norm equations over algebraic number fields by exhaustive search [21], using inequalities given in [20]. In [10], Fieker, Jurk, and Pohst presented an exhaustive search algorithm for solving relative norm equations that uses a modified version of the Finke–Pohst enumeration algorithm of [11]. Finally, Simon developed a way to solve norm equations algebraically using S-units in 2002 [26]. In contrast, solving norm equations in algebraic function fields has undergone far less exploration. The only algorithm in the literature to date, proposed in 2009, is due to Gaál and Pohst [12], who adapted the exhaustive search method of [21] to function fields.

One of the difficulties arising in solving norm equations in any global field is the size of their solutions. An illustrative example of this behaviour is exhibited by algebraic units which can be extremely large, yet their norm is very small. An innovative way to address this problem is to represent the solutions, as well as other field elements arising in the

computation of these solutions, in compact representation. In global function fields, the concept of compact representations was first introduced in 1996 by the third author [23] in the setting of quadratic extensions. In that source and a subsequent paper [24], the concept was used to prove membership in NP and other complexity results for several decision problems arising in computational number theory. In 2013, Eisenträger and Hallgren generalized the concept of compact representation to arbitrary global function fields [9]. They provided a proof that the principal ideal problem in this setting belongs to NP and presented polynomial time quantum algorithms for computing a generator of a principal ideal, the unit group, and the class group of a global function field.

We present two novel algorithms for solving norm equations in global function fields using compact representations. One is an exhaustive search algorithm inspired by the method due to Gaál and Pohst in [12], and the other technique is based on principal ideal testing via index calculus. Our algorithms require the function field to have at least one infinite place of degree 1. We provide detailed complexity analyses of our new algorithms as well as the Gaál–Pohst algorithm [12] and the algorithm for computing compact representations of Eisenträger and Hallgren [9]. All these algorithms were implemented in Magma [7], and we performed extensive numerical experiments to compare their performance in practice. Our Magma implementation, including the testing code, can be found in [17].

Incorporating compact representations into the algorithms not only allowed for smaller representations of solutions, but also led to a significant speed-up compared to the Gaál–Pohst method that does not use compact representations. Our new algorithms outperform the Gaál–Pohst algorithm enormously in terms of run time and have significantly better asymptotic complexities, exponentially better in terms of most of the main function field parameters. The algorithm using index calculus was the most efficient, both in terms of asymptotic complexity and in practice, especially for large inputs.

2 Notation and preliminaries

For background and an algebraic treatment of global function fields, the reader is referred to [22,27].

2.1 Global function fields and norm equations

Let $k = \mathbb{F}_q$ be a finite field of q elements and x be transcendental over k. Denote the polynomial ring and the rational function field over k in x by $k[x]$ and $k(x)$, respectively. A *global function field* F is a finite algebraic extension of $k(x)$. Let n denote its extension degree over $k(x)$, g its genus, and O_F its maximal order, i.e. the integral closure of $k[x]$ in F.

The places of F are partitioned as $P(F) = P_\infty(F) \cup P_0(F)$ where $P_\infty(F)$ is the set of *infinite places* of $F/K(x)$ (consisting of the poles of x), and $P_0(F)$ is the set of *finite places* of $F/k(x)$ (corresponding to the non-zero prime ideals in O_F).

The *maximum norm* of any element $\alpha \in F^\times$ is defined to be

$$\|\alpha\|_\infty = \max_{P \in P_\infty(F)} \{-v_P(\alpha)/e_P\},$$

where $v_P()$ denotes the discrete valuation corresponding to a place $P \in P(F)$ and e_P is the ramification index of P in $F/k(x)$.

Henceforth, we fix a *reduced* basis of $F/k(x)$, i.e. a basis $\mathcal{B} = \{\omega_1, \ldots, \omega_n\}$ that is also a $k[x]$-basis of the maximal order O_F such that $\|\omega_1\|_\infty \leq \cdots \leq \|\omega_n\|_\infty$ and

$$\|\lambda_1\omega_1 + \cdots + \lambda_n\omega_n\|_\infty = \max_{1\leq i\leq n} \|\lambda_i\omega_i\|_\infty$$

for all $\lambda_1, \ldots, \lambda_n \in k(x)$, not all zero.

A *norm equation* in $F/k(x)$ is an identity of the form

$$\mathrm{Norm}_{F/k(x)}(\alpha) = c, \tag{1}$$

where $\alpha \in O_F$ and $c \in k[x] \setminus \{0\}$. Solving a norm equation (1) refers to finding all $\alpha \in O_F$ up to associates, i.e. factors that are units in O_F, such that $\mathrm{Norm}_{F/k(x)}(\alpha) \in ck^\times$. By slight abuse of terminology, a solution of (1) will mean a solution of any of the norm equations $\mathrm{Norm}_{F/k(x)}(\alpha) = \zeta c$ with $\zeta \in k^\times$. Testing whether two elements $\alpha, \beta \in O_F$ are associate is accomplished by identifying all the places $P \in P_0(F)$ for which $v_P(\alpha) \neq 0$ or $v_P(\beta) \neq 0$, and then checking that $v_P(\alpha) = v_P(\beta)$ for all these places.

2.2 *S*-units and their lattices

For any set $S \subseteq P(F)$, let O_S denote the ring of S-integers and $O_S^\times$ the group of S-units. For $S = P_\infty(F)$, we have $O_S = O_F$. Assume now that S is finite with $P_\infty(F) \subseteq S \subset P(F)$. Fixing an order on the places $P_1, \ldots, P_{|S|}$ of S, consider the two group homomorphisms

$$\begin{aligned}
\phi_S &: F^\times \longrightarrow \mathbb{Z}^{|S|}, \qquad \alpha \mapsto \left(-v_{P_1}(\alpha), \ldots, -v_{P_{|S|}}(\alpha)\right), \\
\Phi_S &: F^\times \longrightarrow \mathbb{Z}^{|S|}, \qquad \alpha \mapsto \left(-v_{P_1}(\alpha)\deg P_1, \ldots, -v_{P_{|S|}}(\alpha)\deg P_{|S|}\right).
\end{aligned}$$

The images $\Lambda'_S = \phi(O_S^\times)$ and $\Lambda_S = \Phi_S(O_S^\times)$ are the *S-unit valuation lattice* and the *S-unit lattice* of $F/k(x)$, respectively; they are lattices over $\mathbb{Z}$ of rank $|S|-1$. Put $R'_S = \det \Lambda'_S$ and $R_S = \det \Lambda_S$, where we write $R_S = R_F$ for $S = P_\infty(F)$ and refer to R_F as the *regulator* of $F/k(x)$.

Let $Cl(F)$ denote the class group of F. For any divisor D, let $[D]$ denote its class in $Cl(F)$. Then the kernel of the map

$$\Psi_S : \mathbb{Z}^{|S|} \to Cl(F) \quad \text{via} \quad (v_1, \ldots, v_{|S|}) \mapsto \left[\sum_{i=1}^{|S|} v_i P_i\right] \tag{2}$$

is isomorphic to Λ'_S, so a basis of Λ'_S is obtained as a basis of $\ker(\Psi_S)$, computed in Algorithm 1. Here, we recall that $Cl(F) \cong Cl^0(F) \oplus \mathbb{Z}$ where $Cl^0(F) \subset Cl(F)$ is the degree zero divisor class group of F.

Algorithm 1 SValMat

Input: A finite set $S = \{P_1, \ldots, P_{|S|}\}$ of places including all infinite places of F
Output: An S-unit value matrix $M_S \in \mathbb{Z}^{(|S|-1)\times|S|}$

1: Compute the structure of $Cl^0(F)$ using [15, Algorithmus 5.5] or the technique of [8]
2: $\Psi_S \leftarrow$ the map defined in (2)
3: $\{v_1, \ldots, v_{|S|}\} \leftarrow$ Generators of $\ker(\Psi_S)$
4: $M_0 \leftarrow (|S|-1) \times |S|$ matrix whose i-th row is v_i for $1 \leq i \leq |S|-1$
5: $M_S \leftarrow$ LLL-reduction of M_0
6: **return** M_S

Lemma 2.1 *The cost of Algorithm 1 is heuristically subexponential in g and polynomial in q, n.*

Proof The cost of Algorithm 1 is dominated by step 1. When $g \to \infty$, we use Hess's relation search algorithm [15, Algorithmus 5.5] whose expected run time is subexponential in g under a reasonable smoothness assumption [15, Glattheitsannahme 4.19 and Satz 5.23]. When $q \to \infty$, Diem's method [8] computes $Cl^0(F)$ in heuristic expected time that is polynomial in q. Finally, class group computation is polynomial in n when $n \to \infty$ and g, q are considered fixed. □

For analyzing the size of the output of Algorithm 1, we consider the standard maximum norm of any matrix $M = (m_{ij})$ over $\mathbb{Z}$, given by

$$\|M\|_\infty = \max_{i,j} |m_{ij}|.$$

(This should not be confused with the maximum norm on F; the context makes it clear which norm is under consideration.)

Proposition 2.2 *The output M_S of Algorithm 1 satisfies*

$$\|M_S\|_\infty \leq 2^{(|S|-1)(|S|-2)/4} R'_S.$$

Proof. The rows of M_S form an LLL-reduced basis of Λ'_S. Denoting by $(M_S)_j$ the j-th row of M_S, we have

$$\prod_{j=1}^{|S|-1} \|(M_S)_j\| \leq 2^{(|S|-1)(|S|-2)/4} \det \Lambda'_S = 2^{(|S|-1)(|S|-2)/4} R'_S$$

by [19, Proposition 1.6]. Chose j such that $\|M_S\|_\infty$ is taken on by an element in the j-th row of M_S. Then

$$\|M_S\|_\infty = \|(M_S)_j\|_\infty \leq \prod_{j=1}^{|S|-1} \|(M_S)_j\|_\infty. \quad \square$$

Corollary 2.3 *Let $\mathcal{B}_S = \{b_1, \ldots, b_{|S|-1}\}$ be a LLL-reduced basis of Λ_S. Then*

$$\max_{1 \leq i \leq |S|-1} \|b_i\|_\infty \leq 2^{(|S|-1)(|S|-2)/4} R_S.$$

By [27, Proposition 14.1 (a)], we have $R'_S = |Cl^0(F)/Cl_S(F)| \leq Cl^0(F)$, where $Cl_S(F)$ is the S-class group of F, i.e. the group of divisors supported only outside S modulo principal divisors. Using the bound $|Cl^0(F)| \leq (\sqrt{q}+1)^{2g}$, we obtain

$$R'_S \leq R_F \leq (\sqrt{q}+1)^{2g}. \tag{3}$$

3 Compact representation

An alternative to the standard representation of an element $\alpha \in F$, i.e. given in terms of a $k(x)$-basis of F, is to write α as a tuple of small elements in F such that a suitable power product of these elements evaluates to α. Such a compact representation is particularly well suited for elements α of small norm and a highly useful tool for solving norm equations. We follow the treatment of this subject in [9] and assume throughout this section that $F/k(x)$ has an infinite place of degree 1.

Write $P_\infty(F) = \{P_{\infty,1}, \ldots, P_{\infty,r+1}\}$ with $\deg P_{\infty,r+1} = 1$. It will be helpful to define for any $\alpha \in F$ the r-tuple

$$\mathrm{val}_\infty(\alpha) = (v_{P_{\infty,1}}(\alpha), \ldots, v_{P_{\infty,r}}(\alpha)) \in \mathbb{Z}^r.$$

Let $v = (v_1, \ldots, v_r) \in \mathbb{Q}^r$. Then α is said to be *close* to v if

$$\sum_{i=1}^{r} |v_{P_{\infty,i}}(\alpha) - v_i| \leq r + g.$$

Riemann–Roch spaces and minima play a key role in computing compact representations. The *Riemann–Roch space* of a divisor D of F is the finite dimensional k-vector space

$$L(D) := \{\alpha \in F^\times | \mathrm{div}(\alpha) \geq -D\} \cup \{0\}.$$

For a fractional O_F-ideal I, the *divisor of* I is $\mathrm{div}(I) = \sum_{P \in P_0(F)} n_P P$, where $n_P = v_{\mathfrak{p}}(I)$ and $\mathfrak{p}$ is the O_F-prime ideal corresponding to the place $P \in P_0(F)$. A non-zero element $\mu \in I$ is a *minimum of* I if the following hold. If $\alpha \in I$ is non-zero such that $v_P(\alpha) \geq -v_P(\mu)$ for all $P \in P_\infty(F)$, then either $\alpha = 0$ or $v_P(\alpha) = v_P(\mu)$ for all $P \in P_\infty(F)$. In other words, if $D = \mathrm{div}(I) - \sum_{P \in P_\infty(F)} v_P(\mu) P$, then every $\alpha \in L(D)$ is either 0 or $v_P(\alpha) = v_P(\mu)$ for all $P \in P_\infty(F)$. A fractional O_F-ideal I is *reduced* if 1 is a minimum of I. For any fractional O_F-ideal I and any minimum μ of I, the ideal $(\mu^{-1})I$ is reduced. For a reduced ideal I, we have $0 \leq \deg \mathrm{div}(I) \leq g$.

We now have all the ingredients to introduce compact representations.

Definition 3.1 [9, Definition 4.3] A *compact representation* of $\alpha \in F$ is a pair $\mathbf{t}_\alpha = \big(\mu, (\beta_1, \beta_2, \cdots, \beta_l)\big)$ where

- $l \leq \log\big(\|\mathrm{val}_\infty(\alpha)\|_\infty + g\big)$,
- $\beta_1, \ldots \beta_l \in F$ such that $\beta = \prod_{i=1}^{l} \beta_i^{2^{l-i}}$ is a minimum of O_F,
- $\mu \in F$ satisfies $\alpha = \mu/\beta$,
- The number of bits required to represent μ is polynomial in $\log q$, n and $\deg \mathrm{Norm}_{F/k(x)}(\alpha)$,
- The number of bits required to represent each β_i is polynomial in $\log q$ and n.

This implies in particular that given a compact representation $\mathbf{t}_\alpha = (\mu, (\beta_1, \ldots, \beta_l))$ of $\alpha \in F$, we have

$$\alpha = \mu \prod_{i=1}^{l} \left(\frac{1}{\beta_i}\right)^{2^{l-i}}, \tag{4}$$

where $l, \mu, \beta_1, \ldots, \beta_l$ are all small.

Eisenträger and Hallgen provided an algorithm for computing a compact representation of $\alpha \in F$ in [9] which we reproduce here in more streamlined form. The algorithm first finds μ by computing a basis of a suitable Riemann–Roch space (Algorithm 2) and then computes $\beta_1, \ldots, \beta_l$ via a square-and-multiply approach similar to binary exponentiation (Algorithm 3).

Algorithm 2 Reduce [9, Algorithm 3.4]

Input: A fractional O_F-ideal I and a vector $\nu = (\nu_1, \nu_2, \cdots, \nu_r) \in \mathbb{Z}^r$
Output: A minimum γ of I that is close to ν

```
D ← div(I) + Σ_{i=1}^{r} ν_i P_{∞,i}
for ℓ ∈ [−deg D, −deg D + g] do
    if L(D + ℓP_{∞,r+1}) is not trivial then
        Pick γ from a basis of L(D + ℓP_{∞,r+1})
        return γ
```

To find $\mathbf{t}_\alpha$, we first use Algorithm 2 to find a minimum μ close to 0 in $A = \alpha O_F$. Let $\mathbf{v} = \log_2 \left\| \mathrm{val}_\infty(\mu) - \mathrm{val}_\infty(\alpha) \right\|_\infty$ and put $l = \lfloor \mathbf{v} \rceil + 1$, where $\lfloor \cdot \rceil$ rounds to the nearest integer, rounding up in case of a tie. In the i-th iteration, given $(\beta_1, \ldots, \beta_{i-1})$ from previous iterations, we compute a minimum β_i of the fractional O_F-ideal $\beta_{(i)}^{-2} O_F$ close to $\mathbf{v}/2^{l-i} - 2\,\mathrm{val}_\infty(\beta_{(i)})$, where $\beta_{(i)} = \prod_{j=1}^{i-1} \beta_j^{2^{i-j}}$.

Algorithm 3 CompRep [9, Algorithm 4.6]

Input: A principal ideal $A = \alpha O_F$ and the vector $\mathrm{val}_\infty(\alpha) \in \mathbb{Z}^r$
Output: A compact representation of α

```
μ ← Reduce(A, 0)
l ← ⌊log_2 ‖val_∞(μ) − val_∞(α)‖_∞⌉ + 1
B ← 1 · O_F, β_0 ← 1, ν_β ← 0 ∈ Z^r
for i ∈ {1, ..., l} do
    t ← [⌊(ν_{P_{∞,1}}(μ) − ν_{P_{∞,1}}(α))/2^{l−i}⌉ ... ⌊(ν_{P_{∞,r}}(μ) − ν_{P_{∞,r}}(α))/2^{l−i}⌉]^T
    B ← 1/β_{i−1}^2 B^2
    β_i ← Reduce(B, t − 2ν_β)
    ν_β ← 2ν_β + val_∞(β_i)
return (μ, (β_1, ...β_l))
```

Given the multiplicative structure of compact representations, it is relatively straightforward to devise algorithms for computing products, powers and norms of elements given in compact representation. It is also easy to find the value at any place $P \in P(F)$ of an element in compact representation, ascertain whether an element in compact representation belongs to O_F (by checking that all its values at the finite places are non-negative), and determine whether two elements in compact representation are associate (by comparing their values at all the finite places in their support). We omit the details here and refer to [18, Section 2.4.2] for explicit descriptions of these algorithms.

4 Solving norm equations

In this section, we first describe several techniques for solving norm equations, beginning with the only method found in prior literature, due to Gaál and Pohst [12]. Then we present two new algorithms for accomplishing this task. The method in Sect. 4.2 improves on the exhaustive search approach taken by Gaál–Pohst and incorporates compact representations. Section 4.3 introduces a new algorithm that uses index calculus techniques and also makes use of compact representations.

4.1 Gaál–Pohst

The idea of this method is to look for all non-associate solutions of (1) in a certain region and check that each solution candidate has the correct norm. Since the search space is not explicitly given in [12], we describe it here.

Suppose α is a solution of (1), given in standard representation with respect to a reduced basis $\mathcal{B} = \{\omega_1, \ldots, \omega_n\}$ of $F/k(x)$. Then

$$\alpha = \sum_{i=1}^{n} \lambda_i \omega_i, \tag{5}$$

where $\lambda_i \in k[x]$ for $1 \le i \le n$. Since $\mathcal{B}$ is a reduced basis, we have

$$\|\alpha\|_\infty = \max_{1 \le i \le n} \{\deg \lambda_i + \|\omega_i\|_\infty\},$$

so $\deg \lambda_i \le \|\alpha\|_\infty - \|\omega_i\|_\infty$ for $1 \le i \le n$. Assume that α is minimal among its associate elements with respect to the maximum norm. Then an upper bound on $\|\alpha\|_\infty$ produces a degree bound on λ_i which yields a finite space that we can search for non-associate solutions of (1) in O_F.

Let $\epsilon_1, \ldots, \epsilon_r$ be a system of fundamental units of F and $P \in P_\infty(F)$. For any solution $\beta \in O_F$ of (1), there exist $x_1, \ldots, x_r \in \mathbb{R}$ such that

$$v_P(\alpha) = \sum_{j=1}^{r} x_j v_P(\epsilon_j) + \frac{1}{n} v_P(c) = \sum_{j=1}^{r} x_j v_P(\epsilon_j) - \frac{e_P}{n} \deg c,$$

where e_P is the ramification index of P. This identity is given in [12, p. 244] without proof, but can be derived by adapting the reasoning in [21, Sections 5.3 and 6.4] from number fields to function fields; for details, see [18, Lemma 3.1].

Now put $\alpha = \beta \prod_{j=1}^{r} \epsilon_j^{-\lfloor x_j \rceil}$. Then α is associate to β. Since $1/2 \ge a - \lfloor a \rceil \ge -1/2$ for all $a \in \mathbb{R}$, a simple calculation yields

$$\theta_P - \frac{e_P}{n} \ge v_P(\alpha) \ge -\theta_P - \frac{e_P}{n} \deg c \quad \text{where } \theta_P = \frac{1}{2} \sum_{j=1}^{r} |v_P(\epsilon_j)|. \tag{6}$$

The lower bound implies

$$\|\alpha\|_\infty \le \Theta \quad \text{where } \Theta = \max_{P \in P_\infty(F)} \left\{ \frac{\theta_P}{2e_P} \right\} + \frac{1}{n} \deg c, \tag{7}$$

so

$$\deg \lambda_i \le \Theta - \|\omega_i\|_\infty \quad \text{for } 1 \le i \le n. \tag{8}$$

We can invoke Algorithm 1 to compute the values of a system of fundamental units at the infinite places of F and compute the bounds given in (8) for $1 \le i \le n$. Then we compute the norm of every α of the form (5) such that the coefficients λ_i satisfy (8) and only retain α if its norm is equal to c up to a multiple in $k^\times$. Once all solutions have been found, we remove associate solutions via the procedure described at the end of Sect. 2.1.

4.2 Improved exhaustive search

In this section, we describe a new exhaustive search algorithm for solving norm equations which makes use of compact representations. We assume $\deg P_{\infty,r+1} = 1$. Let

$$S_{c,0} = \{P \in P_0(F) \mid v_P(c) \ne 0\}, \qquad S_c = S_{c,0} \cup P_\infty(F).$$

Then every solution $\alpha \in O_F$ of (1) is an S_c-unit. So if we can bound the values $v_P(\alpha)$ for all $P \in S_c$, then we can search the region defined by these bounds for solutions.

Bounds on $v_P(\alpha)$ for $P \in P_\infty(F)$ are given in (6). Note that the quantities θ_P can easily be obtained from the unit value matrix $M_{P_\infty(F)}$ computed in Algorithm 1. To obtain bounds on $v_P(\alpha)$ for $P \in P_0(F)$, write (1) in the form $\alpha\beta = c$ where $\beta = \mathrm{Norm}_{F/k(x)}(\alpha)/\alpha \in O_F$, which implies $0 \le v_P(\alpha) \le v_P(c)$ for all $P \in P_0(F)$.

We use these bounds to form inputs for Algorithm 3 to compute compact representations of solution candidates of (1). Write

$$S_{c,0} = \{P_1, \ldots, P_{|S_{c,0}|}\}, \qquad P_\infty(F) = \{P_{\infty,1}, \ldots, P_{\infty,r+1}\}.$$

The solutions of (1), up to associates, are in one-to-one correspondence with the principal ideals αO_F dividing cO_F. For $1 \le i \le |S_{c,0}|$, let $\mathfrak{p}_i$ be the prime ideal corresponding to $P_i \in S_{c,0}$. Then all the integral O_F-ideals dividing cO_F are of the form

$$I = \prod_{i=1}^{|S_{c,0}|} \mathfrak{p}_i^{v_i},$$

where $0 \le v_i \le v_{P_i}(c)$ for $1 \le i \le |S_{c,0}|$. If I is principal, say $I = \alpha O_F$, then $v_P(\alpha)$ satisfies the bounds (6) for all $P \in P_\infty(F)$. An additional constraint is given by the fact that the principal divisor of α has degree zero. In other words, we only need to consider tuples $(v_1, \ldots, v_{|S_{c,0}|})$ and $(v_{\infty,1}, \ldots, w_{\infty,r+1})$ such that

$$0 \le v_i \le v_{P_i}(c) \quad \text{for } 1 \le i \le |S_{c,0}|, \tag{9}$$

$$-\theta_{P_{\infty,i}} - \frac{e_{P_{\infty,i}}}{n} \deg c \le v_{\infty,i} \le \theta_{P_{\infty,i}} - \frac{e_{P_{\infty,i}}}{n} \deg c \quad \text{for } 1 \le i \le r+1, \tag{10}$$

$$\sum_{i=1}^{|S_{c,0}|} v_i \deg Q_i + \sum_{i=1}^{r+1} v_{\infty,i} \deg P_{\infty,i} = 0. \tag{11}$$

These conditions are necessary, but not sufficient, for α to be a solution of (1). Nevertheless, the constraint (11) in particular significantly cuts down the number of compact representations that need to be computed.

For every pair (I, V_∞), with $V_\infty = (v_{\infty_1}, \ldots, v_{\infty,r+1})$, that satisfies these conditions, we compute a compact representation $\mathbf{t} = \mathrm{CompRep}(I, V_\infty)$. We then test that $\mathbf{t}$ represents an element in O_F and that this element has the correct norm. We discard $\mathbf{t}$ if it represents an element that is associate to a solution already found. Algorithm 4 shows the whole process.

4.3 Index calculus

In this section, we describe a new exhaustive search algorithm for solving norm equations which also makes use of compact representations, so we assume again that $\deg P_{\infty,r+1} = 1$. Unlike the previous exhaustive search techniques, which enumerate all elements within a large search region, this algorithm enumerates ideals I that divide cO_F and conducts principal ideal tests by solving matrix equations involving a precomputed S-unit value matrix. Using the solutions of the matrix equations, we compute compact representations of solutions of (1).

The solutions α of (1), up to associates, are in bijection with the principal ideals αO_F of norm $c\zeta$ with $\zeta \in k^\times$. By (9), any such ideal must necessarily divide cO_F. So in order to find all solutions, it suffices to consider O_F-ideals I that divide cO_F. If I is principal and has the correct norm, then a generator of I is a solution of (1).

Algorithm 4 Solving (1) via improved exhaustive search

Input: $c \in k[x] \setminus \{0\}$, a reduced basis $\mathcal{B} = \{\omega_i | 1 \le i \le n\}$ of O_F, a unit value matrix $M_{P_\infty(F)} = (m_{i,j})$

Output: A set $\mathcal{R}$ of all non-associate solutions of (1) in compact representation

$\mathcal{R} \leftarrow \emptyset$
$S_{c,0} \leftarrow \{P \in P_0(F) | v_P(c) \neq 0\}$
$v_l \leftarrow \left[\frac{1}{2} \sum_{i=1}^{r} |m_{i,1}| \ \dots \ \frac{1}{2} \sum_{i=1}^{r} |m_{i,r+1}| \right]$
$v_c \leftarrow \left[\frac{e_{P_{\infty,1}}}{n} \deg c \ \dots \ \frac{e_{P_{\infty,r+1}}}{n} \deg c \right]$
for $(v_1, \dots, v_{|S_{c,0}|})$ where $0 \le v_i \le v_{P_i}(c)$ **do**
  for $V_\infty = (v_{\infty,1}, \dots, v_{\infty,r+1})$ where $-(v_l)_i - (v_c)_i \le v_{\infty,i} \le (v_l)_i - (v_c)_i$ **do**
    if $\sum_{i=1}^{|S_{c,0}|} v_i \deg P_i + \sum_{j=1}^{r+1} v_{\infty,j} \deg P_{\infty,j} = 0$ **then**
      $\mathbf{t} \leftarrow \mathrm{CompRep}(I, V_\infty)$
      if $\mathbf{t} \in O_F$ and $\mathbf{t}$ represents an element of norm ζc with $\zeta \in k^\times$ **then**
        if $\mathbf{t}$ represents an element that is not associate to α for any $\mathbf{t}_\alpha \in \mathcal{R}$ **then**
          $\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathbf{t}\}$
          break V_∞
return $\mathcal{R}$

Let S_c and $S_{c,0}$ be as in Sect. 4.2. In order to enumerate all ideals I that divide cO_F, we factor cO_F via a precomputation as

$$cO_F = \prod_{i=1}^{|S_{c,0}|} \mathfrak{p}_i^{v_{P_i}(c)},$$

where for each i, $\mathfrak{p}_i$ is the O_F-prime ideal corresponding to the place $P_i \in S_{c,0}$. Then we perform principal ideal tests on all ideals I dividing cO_F, which are precisely of the form

$$I = \prod_{i=1}^{|S_{c,0}|} \mathfrak{p}_i^{v_{P_i}(I)},$$

where $0 \le v_{P_i}(I) \le v_{P_i}(c)$ for all $1 \le i \le |S_{c,0}|$.

There is a principal ideal test, implemented in Magma, which is an index calculus algorithm that uses Hess's randomized relation search algorithm [15, Algorithmus 5.5]. This algorithm finds a factorization of an ideal equivalent to I by searching relations. When I is principal, the algorithm returns a generator in "factored form". The factored form has subexponentially many terms, each of which has subexponential size in the size of inputs. In our context, the prime ideal factorization of I is already known, and we wish to compute a compact representation of a generator of I if I is principal. Thus, instead of using the existing algorithm, we solve a matrix equation for each I to determine whether or not I is principal and to derive inputs for computing a compact representation.

Let $\{\epsilon_1, \epsilon_2, \dots, \epsilon_{|S_c|-1}\}$ be a system of fundamental S_c-units. Every solution α of (1) is an S_c-unit, so there exist integers $x_i \in \mathbb{Z}$ such that

$$\alpha = \prod_{i=1}^{|S_c|-1} \epsilon_i^{x_i}. \tag{12}$$

We form a matrix $M_{S_{c,0}}$ from the columns of the S_c-value matrix M_{S_c} that correspond to the places in $S_{c,0}$. Here, $M_{S_c} = \mathrm{SValMat}(S_c)$ is precomputed using Algorithm 1.

Now consider the matrix equation

$$\begin{bmatrix} v_{P_1}(\epsilon_1) & v_{P_1}(\epsilon_2) & \dots & v_{P_1}(\epsilon_{|S_c|-1}) \\ v_{P_2}(\epsilon_1) & v_{P_2}(\epsilon_2) & \dots & v_{P_2}(\epsilon_{|S_c|-1}) \\ \vdots & \vdots & \ddots & \vdots \\ v_{P_{|S_{c,0}|}}(\epsilon_1) & v_{P_{|S_{c,0}|}}(\epsilon_2) & \dots & v_{P_{|S_{c,0}|}}(\epsilon_{|S_c|-1}) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{|S_c|-1} \end{bmatrix} = \begin{bmatrix} v_{P_1}(I) \\ v_{P_2}(I) \\ \vdots \\ v_{P_{|S_{c,0}|}}(I) \end{bmatrix}. \tag{13}$$

It is easy to verify that (13) has a solution $\begin{bmatrix} x_1 & \dots & x_{|S_c|-1} \end{bmatrix}^T$ if and only if I is principal, with a generator given by (12). Any such solution gives rise to infinitely many solutions as $M_{S_c,0}$ has fewer rows than columns. However, any two solutions of (13) correspond to associate solutions of (1). So for any O_F-ideal I dividing cO_F, we need only find one solution of (13), and this should be one that gives rise a solution α_0 of (1) whose norm $\|\alpha_0\|_\infty$ is small or even minimal. We proceed as follows.

First, choose a solution $\mathcal{X} = \begin{bmatrix} x_1 & \dots & x_{|S_c|-1} \end{bmatrix}^T$ of (13) that corresponds to $\alpha = \prod_{i=1}^{|S_c|-1} \epsilon_i^{x_i}$. Then compute the vector $v_{\alpha,\infty}$ of the values of α at the infinite places by

$$v_{\alpha,\infty} = M_{S_c,\infty}^T \mathcal{X},$$

where $M_{S_c,\infty}$ is the matrix consisting of the columns of M_{S_c} that are not in $M_{S_c,0}$, i.e. correspond to the infinite places of F. Note that even if $\mathcal{X}$ is short in the Euclidean norm, the quantity $\|v_{\alpha,\infty}\|_\infty$ could still be very large. So we compute a vector v_0 in the unit lattice that is close to $v_{\alpha,\infty}$ with respect to the Euclidean norm and form the vector

$$v_{\alpha_0,\infty} = v_{\alpha,\infty} - v_0.$$

This vector is short in the Euclidean norm and thus corresponds to a generator α_0 of I such that $\|\alpha_0\|_\infty$ is small. From I and $v_{\alpha_0,\infty}$, we can compute a compact representation $\mathbf{t} = \mathrm{CompRep}(I, v_{\alpha_0,\infty})$ of α_0. If it has norm c, it represents a solution of (1).

Algorithm 5 Solving (1) via index calculus

Input: The maximal order O_F of F, $c \in k[x] \setminus k$, an S_c-unit value matrix M_{S_c}
Output: A set $\mathcal{R}$ of all non-associate solutions of (1) that are in O_F in compact representation

$\mathcal{R} \leftarrow \emptyset$
$S_{c,0} \leftarrow \{P \in P_0(F) | v_P(c) \neq 0\}$
$M_{S_c,0} \leftarrow$ the matrix of the columns of M_{S_c} corresponding to the places in $S_{c,0}$
$M_{S_c,\infty} \leftarrow$ the matrix of the columns of M_{S_c} corresponding to the places in $P_\infty(F)$
$M_{P_\infty(F)} \leftarrow$SValMat($P_\infty(F)$)
for every $I | cO_F$ such that $I = \prod_{i=1}^{|S_{c,0}|} \mathfrak{p}^{v_{P_i}(I)}$ **do**
  if $\mathrm{Norm}_{F/k(x)}(I)/c \in k^\times$, **then**
    if (13) is consistent, **then**
      $\mathcal{X} \leftarrow$ a solution of (13) that is short in the Euclidean norm
      $v_0 \leftarrow$ a vector in the lattice generated by the rows of $M_{P_\infty(F)}$, that is closest to $M_{S_c,\infty}^T \mathcal{X}$
      $v \leftarrow M_{S_c,\infty}^T \mathcal{X} - v_0$
      $\mathbf{t} \leftarrow \mathrm{CompRep}(I, v)$
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathbf{t}\}$
return $\mathcal{R}$

In Example 4.1, we compute the search space for each of our three algorithms on input the same norm equation and compare their run times. The computation was performed

with Magma version 2.27-6 on an Intel Xeon CPU E7-8891 v4 with 80 64-bit cores at 2.80 GHz and 256 GB RAM .

Example 4.1 Let $F/\mathbb{F}_5(x)$ be an extension of degree $n = 3$ defined by a root of

$$f(t) = t^3 + (4x^3 + 3x^2 + 1)t^2 + (3x^3 + 4x^2 + 4x + 2)t + 2x^3 + x.$$

Then $F/\mathbb{F}_5(x)$ has two infinite places $P_{\infty,1}$ and $P_{\infty,2}$ with respective ramification indices $e_{P_{\infty,1}} = e_{P_{\infty,2}} = 1$, and hence unit rank $r = 1$.

Let $c = x + 4$. The prime ideal factorization of cO_F is $cO_F = \mathfrak{p}_1\mathfrak{p}_2$, where these two prime ideals correspond to two finite places P_1 and P_2 with $v_{P_1}(c) = v_{P_2}(c) = 1$.

The search space of Gaál–Pohst is determined by the degree bounds in (8). In this example, we have $\deg \lambda_1 \leq 347 + \frac{1}{3}$, $\deg \lambda_2 \leq 344 + \frac{1}{3}$, and $\deg \lambda_3 \leq 344 + \frac{1}{3}$. Thus, a search for solutions requires computing the norms of $5^{348+345+345} = 5^{1038}$ elements in O_F.

The search space of Algorithm 4 is the number of compact representations computed in Step 8, which is the number of tuples satisfying (9), (10), and (11). In our case, we need to find tuples $(v_1, v_2, v_{\infty,1}, v_{\infty,2})$. To satisfy the degree bound in (11), we only need to choose the first 3 numbers to determine the tuple. Since we have $0 \leq v_1, v_2 \leq 1$ and $-347 - \frac{1}{3} \leq v_{\infty_1} \leq 347 - \frac{1}{3}$, we have up to $2 \cdot 2 \cdot (347 \cdot 2 + 1) = 2980$ possible tuples which is much less than the search bound for Gaál–Pohst.

Lastly, the number of ideals to enumerate in Algorithm 5 is the number of pairs (v_1, v_2). With the same bounds on v_1, v_2 above, we only need to search 4 ideals which is significantly less than the previous two algorithms.

The search of Gaál–Pohst did not finish within 4 days, so we terminated the computation. Our improved exhaustive search algorithm took 114.83 CPU seconds, and the index calculus algorithm only took 0.18 CPU seconds for the entire process.

5 Complexity analysis

In this section, we analyze the complexity of the compact representation and norm equation algorithms. Throughout, $F/k(x)$ is represented by a monic irreducible polynomial $f(t) = t^n + a_{n-1}t^{n-1} + \cdots + a_0 \in k[x][t]$. The size of this representation is captured by the quantity

$$C_f = \max\left\{ \left\lceil \frac{\deg a_i(x)}{i} \right\rceil \,\middle|\, 1 \leq i \leq n \right\}. \tag{14}$$

Note that $C_f = O(g)$ when $g \to \infty$; see [2, Corollary 3.5]. Except for the original Gaál-Pohst method, we assume that $F/k(x)$ has a place of degree one.

Our asymptotic run times count bit operations and are expressed as functions of $q = |k|$, $n = [F : k(x)]$, g (the genus of F), and the sizes of the inputs specific to each algorithm. Some complexity estimates include other quantities, such as the regulator R_F or the unit rank r of $F/k(x)$. If all these quantities are present, $O()$ constants should be understood as true constants. Later, we will consider asymptotics where one of q, n, g or [in the case of solving norm equations (1)] $\deg c$ grows and the other quantities are assumed to be fixed; these $O()$ constants will then depend on the fixed parameters. For any quantity X, we simplify any power of $\log X$ to writing X^{ε}.

For basic arithmetic ingredients, we assume the following complexities:

- Multiplication of two elements in k: $O((\log q)^{1+\varepsilon}) = O(q^{\varepsilon})$ [14];

- Multiplication of two polynomials in $k[x]$ of degree d: $O(d^{1+\varepsilon}q^{\varepsilon})$ [14];
- Computing the determinant of a matrix $M = (m_{ij}) \in k[x]^{n\times n}$: $O(n^{\omega+\varepsilon}q^{\varepsilon}\lceil s(M)\rceil)$, where $\omega < 2.37286$ (see [16, Proposition 3.3] and [1]) and

$$s(M) = \frac{1}{n}\sum_{i=1}^{n}\left(\max_{1\le j\le n}(\deg M)_{i,j}\right)$$

  is the average column degree of M.
- Factoring a polynomial in $k[x]$ of degree d: $O(d^{\omega}q^{\varepsilon})$ using Berlekamp's algorithm [29, Theorem 14.32].

Sizes of elements are measured in *heights* which are defined as follows:

- For $\lambda = \lambda_1/\lambda_2 \in k(x)$ with coprime polynomials $\lambda_1, \lambda_2 \in k[x]$, $\lambda_2 \neq 0$, we define $\mathbf{h}(\lambda) = \max\{\deg\lambda_1, \deg\lambda_2\}$.
- For a $k(x)$-basis $\mathcal{B} = \{\omega_1, \ldots, \omega_n\}$ of F and $\alpha = \sum_{i=1}^{n}\lambda_i\omega_i \in F$ with $\lambda_i \in k(x)$ for $1 \le i \le n$, we define $\mathbf{h}_{\mathcal{B}}(\alpha) = \max_{1\le i\le n}\mathbf{h}(\lambda_i)$.
  For a precomputed fixed reduced basis $\mathcal{B}$, we write $\mathbf{h}(\alpha)$ for $\mathbf{h}_{\mathcal{B}}(\alpha)$.
- For a divisor $D = \sum_P n_P P$ of F, we define $\mathbf{h}(D) = \sum_P |n_P|\deg P$.

We assume that we have precomputed a reduced basis $\mathcal{B} = \{\omega_1, \ldots, \omega_n\}$ of $F/k(x)$ and polynomials $a_{ijm} \in k[x]$ such that $\omega_i\omega_j = \sum_{m=1}^{n} a_{ijm}\omega_m$ for $1 \le i, j \le n$. The elements of a reduced basis are short; specifically

$$\|\omega_n\|_{\infty} \le \left\lceil \frac{2g-1}{n} \right\rceil + 1 \tag{15}$$

by [28, Theorem 5.4.1].

Fractional O_F-ideals I are given in *Hermite Normal Form* (HNF) representation, i.e. as a pair $(M_I, d(I))$. Here, $d(I)$ is the denominator of I, i.e. the monic polynomial $d \in k[x]$ of minimal degree such that $dI \subseteq O_F$, and M_I is the coefficient matrix of a $k[x]$-basis of $d(I)I$ in HNF.

The next two lemmas provide the cost of norm computation.

Lemma 5.1 *For $\alpha \in F$, computing $\mathrm{Norm}_{F/k(x)}(\alpha)$ requires $O(n^3 d_{\alpha}^{1+\varepsilon}q^{\varepsilon})$ bit operations, where $d_{\alpha} = \max\left\{\mathbf{h}(\alpha), 2\left(\left\lceil\frac{2g-1}{n}\right\rceil + 1\right)\right\}$.*

Proof We have $\mathrm{Norm}_{F/k(x)}(\alpha) = \det(M_{\alpha})$, where $M_{\alpha} \in k(x)^{n\times n}$ is the unique matrix such that $\alpha\,[\omega_1, \ldots, \omega_n] = [\omega_1, \ldots, \omega_n]\,M_{\alpha}$. Writing $\alpha = \sum_{i=1}^{n}\lambda_i\omega_i$ and $M_{\alpha} = (m_{ij})$, we have $m_{jl} = \sum_{i=1}^{n}\lambda_i a_{ijl}$ and $\deg m_{jl} \le 2d_{\alpha}$ by (15). So computing M_{α} takes $O\left(n^3 d_{\alpha}^{1+\varepsilon}q^{\varepsilon}\right)$ bit operations, and this dominates the cost of computing $\det M_{\alpha}$. □

Lemma 5.2 *For a fractional O_F-ideal I in HNF representation, computing $\mathrm{Norm}(I)$ requires $O\left(\left(n^{2+\varepsilon}\deg\mathrm{Norm}(d(I)I) + (n\deg d(I))^{1+\varepsilon}\right)q^{\varepsilon}\right)$ bit operations.*

Proof Let $(M_I, d(I))$ be the HNF representation of I. Then $\mathrm{Norm}(I) = \det M_I/d(I)^n$. By [28, Proposition 5.1.17], we have $\|M_I\|_{\infty} \le \deg\mathrm{Norm}(d(I)I)$. So the cost of computing $\det M_I$ is $O(n^{2+\varepsilon}\deg\mathrm{Norm}(d(I)I)q^{\varepsilon})$, and that of computing $d(I)^n$ is $O((n\deg d(I))^{1+\varepsilon}q^{\varepsilon})$. □

5.1 Compact representation

The cost of computing compact representations using Algorithm 3 is dominated by the calls to Algorithm 2 whose cost in turn is dominated by computing k-bases of at most $g+1$ Riemann–Roch spaces. We assume that $\deg P_{\infty,r+1} = 1$. Let D be a divisor of F. For brevity, we denote the cost of computing a k-basis of the Riemann–Roch space $L(D)$ by $\mathsf{RR}(\mathbf{h}(D))$. By [3, Theorem 4.13], we have

$$\mathsf{RR}(\mathbf{h}(D)) = O\left(\left(n^5(\mathbf{h}(D)+n^2C_f)^2 + n^{5+\varepsilon}C_f^{2+\varepsilon}\right)q^{\varepsilon}\right) \tag{16}$$

bit operations, with C_f as in (14), and $\mathsf{RR}(m\mathbf{h}(D)) = \mathsf{RR}(\mathbf{h}(D))$ for $m \in \mathbb{R}$.

Lemma 5.3 *Let I be a fractional ideal and $v \in \mathbb{Z}^r$. On input I and v, Algorithm 2 requires $O\big(g\,\mathsf{RR}(n\mathbf{h}(\mathrm{Norm}(I)) + n\,\|v\|_\infty + g)\big)$ bit operations.*

Proof For the divisor D formed in step 1, we have $\mathbf{h}(D) \le \mathbf{h}(\mathrm{div}(I)) + n\,\|v\|_\infty$, where

$$\mathbf{h}(\mathrm{div}(I)) \le \sum_{P\in P_0(F)} |\nu_P(\mathrm{div}(I))| \deg P \le \sum_{P\in P_0(F)} |\nu_P(\mathrm{Norm}(I))| \deg P \le 2n\mathbf{h}(\mathrm{Norm}(I)).$$

The last inequality can be obtained from the factorization of $\mathrm{Norm}(I)$ into irreducible polynomials in $k[x]$; see [18, Lemma 2.35].

The interval containing ℓ in step 2 forces $|\ell| \le h(D)+g$, so

$$\mathbf{h}(D+\ell P_{r+1}) \le 2h(D)+g \le 2n\mathbf{h}(\mathrm{Norm}(I)) + n\,\|v\|_\infty + g.$$

The loop in step 2 is executed $g+1$ times, so the result follows from (16). □

Lemma 5.4 *Let A be a principal O_F-ideal in HNF-representation, generated by an element $\alpha \in F$. On input A and $\mathrm{val}_\infty(\alpha)$, Algorithm 3 requires*

$$O\Big(g\big(\mathsf{RR}(n\mathbf{h}(\mathrm{Norm}_{F/k(x)}(\alpha)) + g) + \log\left(\left\|\mathrm{val}_\infty(\alpha)\right\|_\infty + g\right)\mathsf{RR}(n^2+ng)\big)\Big)$$

bit operations.

Proof. We use the fact that $\beta = \mu/\alpha$ is a minimum of O_F, so $B = (\beta^{-1})O_F$ is a reduced O_F-ideal, and hence $0 \le \deg(\mathrm{div}(B)) \le g$. By Lemma 5.3, the cost of step 1 is $O\left(g\,\mathsf{RR}(n\mathbf{h}(\mathrm{Norm}_{F/k(x)}(\alpha)) + g)\right)$ bit operations. Similar reasoning shows that the cost of computing each β_i in step is $O\left(g\,\mathsf{RR}(n^2+ng)\right)$ bit operations. The number l of loop iterations defined in step 2 can be bounded by

$$l = \lfloor \log_2(\left\|\mathrm{val}_\infty(\beta)\right\|\rceil + 1 \le \lfloor \log_2(\left\|\mathrm{val}_\infty(\alpha)\right\|_\infty + g)\rceil + 1.$$ □

Let D be any divisor of F and $\alpha \in L(D)$. Then [4, Lemma 3.5] implies that

$$\mathbf{h}(\alpha) = O(\mathbf{h}(D)+n). \tag{17}$$

We can now bound the heights of the quantities comprising a compact representation.

Lemma 5.5 *Let $\mathbf{t}_\alpha = (\mu, \beta_1, \ldots, \beta_l) = CompRep(\alpha O_F, v_\infty(\alpha))$ be a compact representation of $\alpha \in F$, and let C_f be as defined in (14). Then the following hold.*

$$\begin{aligned} l &= O(\log\left\|v_\infty(\alpha)\right\|_\infty + g),\\ \mathbf{h}(\mu) &= O(n\mathbf{h}(\mathrm{Norm}_{F/k(x)}(\alpha)) + g + n),\\ \mathbf{h}(\beta_i) &= O(n^2+ng) \ \textit{for}\ 1 \le i \le l. \end{aligned}$$

Proof The bound on l was established in the proof of Lemma 5.4. We have $\mu \in L(D)$ where $D = \operatorname{div}(\alpha O_F) + \ell P_{r+1}$. The bound on $\mathbf{h}(\mu)$ follows from the bound on $\mathbf{h}(D)$ given in the proof of Lemma 5.4 and (17).

By [9, Proof of Proposition 4.11], each β_i is a minimum in a fractional O_F-ideal B_i^2 close to a vector $t_i = (t_{i1}, \ldots, t_{ir})$, where B_i is a reduced ideal and $\|t_i\|_\infty = O(n+g)$. So $\beta_i \in L(D_i)$ where

$$\mathbf{h}(D_i) = \mathbf{h}(\operatorname{div}(B_i^2)) + \mathbf{h}\left(\sum_{j=1}^{r} t_{ij} P_j + \ell P_{r+1}\right) = O(g + n\,\|t_i\|_\infty) = O(n^2 + ng).$$

Thus, $\mathbf{h}(\beta_i) = O(n^2 + ng)$ by (17). □

5.2 Gaál–Pohst

The cost of the Gaál–Pohst method [12] is dominated by computing the norms of elements in the search space and checking whether solutions are associate. By (8), the number of elements in the search space is bounded by

$$\prod_{i=1}^{n} q^{\lfloor \Theta - \|\omega_i\|_\infty \rfloor + 1} \le q^{n\Theta}, \tag{18}$$

with Θ given by (7).

To test associateness of two elements, we factor the principal ideals they generate, which is accomplished by factoring their norms using Berlekamp's algorithm. This yields the following complexity for Gaál–Pohst's exhaustive search method.

Theorem 5.6 *Let $T = r2^{r(r-1)/4-1}R_F + \frac{1}{n}\deg c$, where R_F is the regulator and r the unit rank of $F/k(x)$. Let $d_T = \max\{T, \lceil \frac{2g-1}{n} \rceil\}$. Then the Gaál–Pohst method can solve Eq. (1) in*

$$O\left(2^{nTq^\varepsilon} q^\varepsilon (n^3 d_T^{1+\epsilon} + (\deg c)^\omega)\right)$$

bit operations, when n, g, q and $\deg c \to \infty$.

Proof By (8), we have $\mathbf{h}(\alpha) \le \Theta$ for every α in the search space. By Corollary 2.3, we have $\Theta \le T$. Thus, computing the norm of each α can be done in time $O(n^3 d_T^{1+\epsilon} q^\varepsilon)$ by Lemma 5.1. By (18), we compute the norms of up to $q^{n\Theta}$ elements. The cost of testing whether two solutions of (1) are associate is $O((\deg c)^\omega q^\varepsilon)$ via norm factorization, and the number of tests that need to be performed is bounded above by $q^{n\Theta}$. □

We briefly discuss the asymptotic complexity and the sizes of the solutions α produced by the Gaál–Pohst method in the different asymptotic settings where one of n, g, $\deg(c)$, q tends to infinity and the other three quanities are fixed. By (7), we have $\mathbf{h}(\alpha) \le \Theta \le T$. We note that $r \le n$ and use (3) to bound R_F.

- $n \to \infty$: run time $2^{O(n^2 2^{n^2/4})}$, $\mathbf{h}(\alpha) = O(n2^{n^2/4})$;
- $g \to \infty$: run time $2^{O(q^g)}$, $\mathbf{h}(\alpha) = 2^{O(g)}$;
- $\deg c \to \infty$: run time $O\left(2^{q^\varepsilon \deg c}(\deg c)^\omega\right)$, $\mathbf{h}(\alpha) = O(\deg c)$;
- $q \to \infty$: run time $q^{O(q^g)}$, $\mathbf{h}(\alpha) = O(q^g)$.

5.3 Improved exhaustive search

The cost of Algorithm 4 is dominated by computing compact representations and their norms. We assume that F has an infinite place of degree 1, labelled as before by $P_{\infty,r+1}$.

The number of compact representations computed in Algorithm 4 is bounded by the number of tuples $(v_1, \ldots, v_{|S_{c,0}|}, v_{\infty,1}, \ldots, w_{\infty,r+1})$ that satisfy (9) and (10). The number of $v_1, \ldots, v_{|S_{c,0}|}$ satisfying (9) is

$$\prod_{i=1}^{|S_{c,0}|} (v_{P_i}(c) + 1). \tag{19}$$

Similarly counting the number of $(v_{\infty,1}, \ldots, v_{\infty,r+1})$ that satisfy (10), we obtain an upper bound of

$$\prod_{i=1}^{|S_{c,0}|} (v_P(c) + 1) \prod_{j=1}^{r} (2\theta_{\infty,j} + 1) \tag{20}$$

on the number of compact representations computed in Algorithm 4. This yields the following cost estimate for Algorithm 4.

Theorem 5.7 *With $T' = r2^{r(r-1)/4-1}R_F$, Algorithm 4 can solve Eq. (1) in*

$$O\Bigg(\Big(g\mathsf{RR}(n\deg(c)+g) + g\log(T'+g)\mathsf{RR}(n^2+ng) + n^{5+\varepsilon}(T'\deg c)^{1+\varepsilon} + (\deg c)^{\omega}\Big) \times q^{\varepsilon}(2T'+1)^r \prod_{P\in S_{c,0}} (v_P(c)+1)\Bigg)$$

bit operations, when n, g, q and $\deg c \to \infty$.

Proof The number of compact representations computed in Algorithm 4 is given in (20). By Lemma 5.4, computing each compact representation takes

$$O\Big(gq^{\varepsilon}\big(\mathsf{RR}(n\deg(c)+g) + \log(\max_i \theta_{\infty,i} + g)\mathsf{RR}(n^2+ng)\big)\Big)$$

bit operations. By Corollary 2.3, we have $\theta_{\infty,i} \le \Theta \le T'$ for $1 \le i \le r+1$. The cost of testing whether or not such a compact representation is in O_F is $O((\deg c)^{\omega}q^{\varepsilon})$. For each compact representation computed in step 8, we have $l = O\left(\log \left\| M^T_{S_c,\infty} \right\|_{\infty}\right)$. From Lemma 5.5, we also have $\mathbf{h}(\beta_i) = O(n^2 + ng)$ for $1 \le i \le l$ and $\mathbf{h}(\mu) = O(n\mathbf{h}(\mathrm{Norm}(I)) + g + n = O(n\deg c + g + n)$ because I divides cO_F. By [18, Lemma 2.48], the cost of computing the norm of such a compact representation is thus $O(n^{5+\varepsilon}(T'\deg c)^{1+\varepsilon}q^{\varepsilon})$. Finally, the cost of testing associateness of any two such compact representations is again the same as that of factoring cO_F, i.e. $O((\deg c)^{\omega}q^{\varepsilon})$. □

Again we analyze the complexity of Algorithm 4 under different asymptotic assumptions, with one of $n, g, \deg c, q$ tending to infinity and the others remaining fixed. To simplify the expression in Theorem 5.7, we bound (19). This quantity varies greatly depending on the factorization of c. It takes on its minimal possible value when cO_F has only one unramified (i.e. inert) place P, in which case $v_P(c) = 1$. Its maximum occurs when c splits into linear factors and each linear factor splits completely, in which case $v_P(c) = 1$ for all $P \in S_{c,0}$ and $|S_{c,0}| = n\deg c$. Thus,

$$2 \le \prod_{P\in S_{c,0}} (v_P(c)+1) \le 2^{n\deg c}. \tag{21}$$

Along with these bounds, we again use $r \le n$ and bound R_F via (3).

5.3.1 Case $n \to \infty$

Here, $T' = O(n2^{n^2/4})$. Using the upper bound in (21), we obtain $|S_c| = O(n)$. If cO_F is a prime ideal, in which case the lower bound in (21) applies, the asymptotic run time improves by a factor of 2^n. In this case, $\mathbf{h}(\mathrm{div}(cO_F)) = O(n)$. By Proposition 2.2 and (7), the size of a solution in compact representation in Algorithm 4 is polynomial in n.

5.3.2 Case $g \to \infty$

Here $T' = O(2^{O(g)})$ and $\mathsf{RR}(n^2 + ng) = O(g^{2+\varepsilon})$ by (16) as $C_f = O(g)$, yielding an asymptotic run time of $O\left(g^{4+\varepsilon}2^{g(n+1)q^\varepsilon}\right) = 2^{O(g)}$ for Algorithm 4. Compact representations of solutions have polynomial height in g.

5.3.3 Case $\deg c \to \infty$

The worst case is again given by the upper bound in (21), in which vase $|S_c| = O(\deg c)$. From (16), we see that $\mathsf{RR}(n \deg c + g) = O((\deg c)^2)$, giving an asymptotic complexity of $O\left(2^{n \deg c}(\deg c)^\omega\right)$ for Algorithm 4. When cO_F is a prime ideal, this improves to $O\left((\deg c)^\omega\right)$, which is polynomial in $\deg c$. Here, $\mathbf{h}(\mathrm{div}(cO_F)) = O(\deg c)$, so a solution in compact representation has polynomial size in $\deg c$.

5.3.4 Case $\deg q \to \infty$

Here, (3) yields an asymptotic complexity of $O(q^{gn+\varepsilon})$ for Algorithm 4.

Note that in all cases, Algorithm 4 represents an enormous speed-up over Gaál–Pohst, and the solutions have far smaller sizes. So introducing compact representation results in a significant gain in time and space efficiency.

5.4 Index calculus

We now analyze the expected running time of Algorithm 5. Again we assume that F has at least one infinite place of degree 1. Define

$$d_{S_c} := \max\left\{2^{(|S_c|-1)(|S_c|-2)/4}R'_S,\ \max_{P\in S_{c,0}} \nu_P(c)\right\}. \tag{22}$$

For any augmented matrix $\left[M_{S_c}^T \,\middle|\, V\right]$ in (13), we have $\left\|\left[M_{S_c}^T \,\middle|\, V\right]\right\|_\infty \le d_{S_c}$, and $\max_{1\le i\le |S_c|-1}\{\mathbf{h}(\epsilon_i)\} \le d_{S_c}$. Note that d_{S_c} only depends on c and F.

We enumerate all O_F-ideals I that divide cO_F as given in (9). The number of these I is given by (19). For each such I, we must compute five components: the HNF representation of I, the norm $\mathrm{Norm}_{F/k(x)}(I)$, a solution $\mathcal{X}$ of (13), a vector v_0 in the unit lattice that is close to $\mathcal{X}$, and a compact representation of the solution of (1) corresponding to $\mathcal{X}$.

Computing the HNF representation of such I entails computing products of HNF representations which, by [28, Theorem 5.2.2] for example, can be done in $O\left(n^7g^2q^\varepsilon\right)$ bit operations.

Since c is in $k[x]$, each I is integral, so Lemma 5.2 shows that computing the norm of each I is $O\left(\left(n^{2+\varepsilon}\deg c + (n\deg c)^{1+\varepsilon}\right)q^\varepsilon\right)$ bit operations.

To determine the cost of solving a matrix equations $M_{S_c,0}^T X = V$ as given in (13), we note that

$$\left\|M_{S_c,0}^T\right\|_\infty = \max_{\substack{P\in P_\infty(F)\\ 1\le i\le |S_c|-1}} \nu_P(\epsilon_i), \qquad \|V\|_\infty \le \max_{P\in P_0(F)} \nu_P(c).$$

by construction. By [6, Theorem 22], the cost of solving $M_{S_c,0}^T X = V$ is $O(|S_c|^{\omega+\epsilon} d_{S_c}^{\epsilon})$ bit operations.

To find a vector v_0 in the unit lattice closest to $M_{S_c,\infty}^T \mathcal{X}$, we can use the algorithm in [5]. Its expected run time is $O(2^{0.3774r})$, and we have

$$\left\| M_{S_c,\infty}^T \mathcal{X} - v_0 \right\|_\infty \leq \max_{1\leq i\leq r} \left\{ \sum_{j=1}^{r} \frac{1}{2} |v_{P_i}(\epsilon_j)| \right\} \leq d_{S_c}.$$

Finally, computing a compact representation $\mathbf{t}$ of a solution corresponding to $\mathcal{X}$ is done by invoking Algorithm 3 on inputs $A = \prod_{i=1}^{|S_{c,0}|} \mathfrak{p}_i^{(V)_i}$ and $V - v_0$, where $\mathfrak{p}_i$ is the prime ideal corresponding to the place $P_i \in S_{c,0}$. We have $\mathbf{h}(\mathrm{div}(A)) \leq \mathbf{h}(\mathrm{div}(cO_F))$ and $\mathbf{h}(\mathrm{Norm}(A)) = \deg c$. Thus, computing $\mathbf{t}$ takes

$$O\Big(gq^{\varepsilon}\left(\mathsf{RR}(n\deg c + g) + \log\left(d_{S_c} + g\right)\mathsf{RR}(n^2 + ng)\right)\Big)$$

bit operations by Lemma 5.4.

Putting all these estimates together, we obtain the following asymptotic run time for Algorithm 5.

Theorem 5.8 *With d_{S_c} defined as in (22), Algorithm 5 requires*

$$O\Bigg(\Big(\prod_{P\in S_{c,0}} (v_P(c)+1)\Big)\Big(|S_c|^{\omega+\varepsilon} d_{S_c}^{\varepsilon} + 2^{0.3774r} + q^{\varepsilon}\left(n^7 g^2 + n^{2+\varepsilon}(\deg c)^{1+\varepsilon}\right.$$
$$\left.+g\mathsf{RR}(n\deg c + g) + g\log(d_{S_c}+g)\mathsf{RR}(n^2+ng)\right)\Big)\Bigg)$$

bit operations to solve a norm equation (1) when n, g, $\deg c$, and $q \to \infty$.

Proof Each iteration of the loop in steps 6 consists of the five components listed above for an O_F-ideal I, and thus takes

$$O\Big(\left(n^7 g^2 + n^{2+\varepsilon}\deg c + (n\deg c)^{1+\varepsilon}\right) q^{\varepsilon} + |S_c|^{\omega+\varepsilon} d_{S_c}^{\varepsilon} + 2^{0.3774r}$$
$$+gq^{\varepsilon}\left(\mathsf{RR}(n\deg c + g) + \log(d_{S_c}+g)\mathsf{RR}(n^2+ng)\right)\Big)$$

bit operations. The number of iterations is the quantity in (19). Combining all these costs yields the result. □

The compact representations produced by Algorithm 5 are subject to the same height bounds as those obtained from Algorithm 4.

We consider the complexity of Algorithm 5 when, as before, only one of n, g, $\deg c$ and q tends to infinity and the others stay fixed. For a bound on d_{S_c} as given in (22), we bound R_F by (3) and use (21).

5.4.1 Case $n \to \infty$

From (16), we obtain $\mathsf{RR}(n^2 + ng) = O(n^9)$. The upper and lower bounds in (21) yield $d_{S_c} = O(2^{n^2})$ and $d_{S_c} = O(1)$, respectively. The asymptotic complexity of Algorithm 5 is $O\left(2^{n(0.3774+\deg c)}\right) = 2^{n(\deg c + o(1))}$ and $O\left(2^{0.3774n}\right)$, respectively, i.e. exponential. In the first case, this is due to the fact that the number of ideals diviing cO_F is exponential in n, whereas in the second case, the estimate arises from the cost of solving the system (13).

Table 1 Summary of asymptotic complexity of Gaál–Pohst, Algorithms 4 and 5

Asymptotic	Gaál–Pohst	Algorithm 4	Algorithm 5
$n \to \infty$ (worst)	$2^{O(n^2 2^{n^2/4})}$	$2^{n^3(1+o(1))}$	$2^{n(\deg c+o(1))}$
$n \to \infty$, cO_F prime (best)	$2^{O(n^2 2^{n^2/4})}$	$2^{n^3(1+o(1))}$	$O(2^{0.3774n})$
$g \to \infty$	$2^{O(q^g)}$	$2^{O(g)}$	$O(g^{4+\varepsilon})$
$\deg c \to \infty$ (worst)	$O\left(2^{q^\varepsilon \deg c}(\deg c)^\omega\right)$	$O\left(2^{n \deg c}(\deg c)^\omega\right)$	$O\left(2^{n \deg c}(\deg c)^{\omega+2+\varepsilon}\right)$
$\deg c \to \infty$, cO_F prime (best)	$O\left(2^{q^\varepsilon \deg c}(\deg c)^\omega\right)$	$O\left((\deg c)^\omega\right)$	$O\left((\deg c)^2\right)$
$q \to \infty$	$q^{O(q^g)}$	$O(q^{gn+\varepsilon})$	$O(q^\varepsilon)$

5.4.2 Case $g \to \infty$

Here, $\mathrm{RR}(n^2+ng) = O(g^{2+\varepsilon})$, and $\log(d_{S_c}+g) = O(g^{1+\varepsilon})$, giving a polynomial asymptotic complexity of $O\left(g^{4+\varepsilon}\right)$ for Algorithm 5. This is asymptotically less than the precomputation of M_{S_c}, which is subexponential in g by Lemma 2.1.

5.4.3 Case $\deg c \to \infty$

From (16), we obtain $\mathrm{RR}(n \deg c + g) = O((\deg c)^2)$. The bounds on d_{S_c} are the same as in the case $n \to \infty$. The upper and lower bounds in (21) yield respective asymptotic complexities of $O\left(2^{n \deg c}(\deg c)^{\omega+2+\varepsilon}\right)$ and $O((\deg c)^2)$ when cO_F is a prime ideal for Algorithm 5; in the latter case, the cost of computing compact representations dominates the overall run time. Note that factoring the ideal cO_F, performed as a precomputation, is actually asymptotically more costly than the algorithm itself when cO_F is prime.

5.4.4 Case $q \to \infty$

Here, the asymptotic run time of Algorithm 5 is $O\left(q^\varepsilon\right)$ which is less expensive than the precomputation of M_{S_c}, whose cost is given in Lemma 2.1.

Once again, in all cases Algorithm 5 substantially outperforms Algorithm 4, and Gaál–Pohst even more so. This is because the number of ideals tested in Algorithm 5 is far smaller than the number of elements in the search spaces of the other two algorithms.

Table 1 summarizes the asymptotic complexities of the Gaál–Pohst method, Algorithms 4 and 5. The complexity estimates in this table are born out by our numerical experiments, described in the next section.

6 Empirical analysis

The Gaál–Pohst method, Algorithms 4 and 5 were all implemented in Magma [7], version 2.27-6. Our code is available on the first author's GitHub [17]. All experiments were performed on an Intel Xeon CPU E7-8891 v4 with 80 64-bit cores at 2.80 GHz and 256 GB RAM. Selected test results are provided in Figs. 1 and 2; an extensive suite of tests and their results can be found in [18, Chapter 5].

All tests and timings were performed on function fields $F/\mathbb{F}_q(x)$, where q is a prime not dividing $n = [F : k(x)]$ and F has at least one infinite place of degree 1. The fields in the selected results presented here all had ideal class number $h_{O_F} = 1$. Tests were conducted with randomly generated function fields with specified n, g, q, $\deg c$. Computations were forcefully terminated when the CPU time required for an algorithm and its precomputation exceeded one day. Timings are given in terms of the average number of CPU seconds.

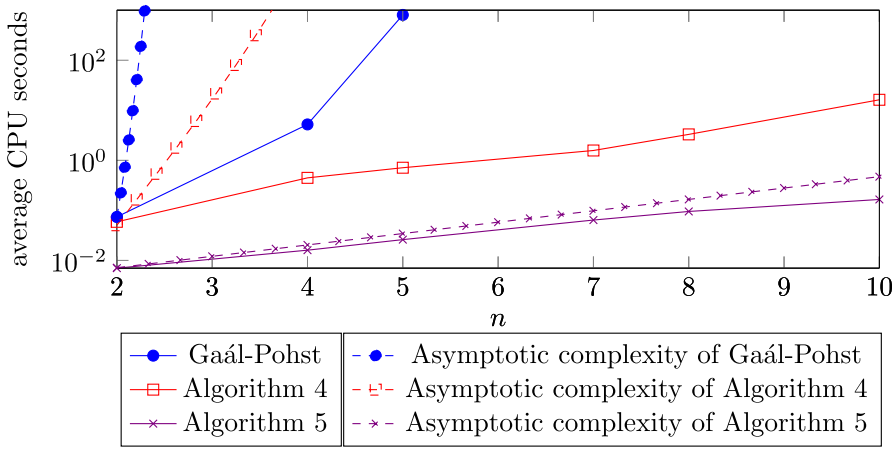

Fig. 1 Empirical timing results and asymptotic complexities for varying n ($g = 1, q = 3, h_{O_F} = 1, \deg c = 1$, and irreducible c)

We summarize our observations of our timing tests as follows:

- In all test cases, for any norm equation, Algorithm 5 outperformed Algorithm 4 and the Gaál–Pohst method.
- Our timing results are largely well-aligned with our complexity analysis. Since we only considered worst case complexities, they are not expected to match exactly.
- With a set of minimal parameters, $g = 1$, $q = 3$, $n = 1$, $\deg c = 1$, the Gaál–Pohst method was as fast as Algorithm 4.
- As expected from the complexity results, Gaál–Pohst slowed down at a much faster rate than the other two algorithms as the parameter sizes increased. Figure 1 illustrates this.
- The average CPU time taken for precomputing an S-unit value matrix was negligible compared to Gaál–Pohst and Algorithm 4 in all test examples. However, the precomputation took longer than Algorithm 5 in many cases. This is also not unexpected.

Figure 1 shows the timing results with n varying from 1 to 10 and $g = 1$, $q = 3$, $\deg c = 1$ fixed, along with the asymptotic complexities. The timing data of Gaál–Pohst are only available for $n = 2, 4, 5$, because for $n = 7$, solving one norm equation already took more than a day due to the huge search space. As expected, the shapes of the graphs of the asymptotic complexities and timing results are largely similar, but the actual run times grow slower than the asymptotic complexities.

For all the testing examples, Algorithm 5 was the fastest. Gaál–Pohst was faster than Algorithm 4 for some examples with $n = 2$, but on average Algorithm 4 was faster. From $n = 3$ on, Algorithm 4 outperformed Gaál–Pohst significantly. Gaál–Pohst and Algorithm 4 are more substantially affected by the growth in n, mainly because their search spaces expand doubly exponentially as n grows, while the number of ideals to search in Algorithm 5 grows exponentially.

Figure 2 shows the timing results for varying g with $1 \leq g \leq 18$, with $n = 2$, $q = 3$, $\deg c = 1$. Again, there is good agreement between empirical and predicted run times. No timings for Gaál–Pohst results are shown because the test took too long. For $g = 1$,

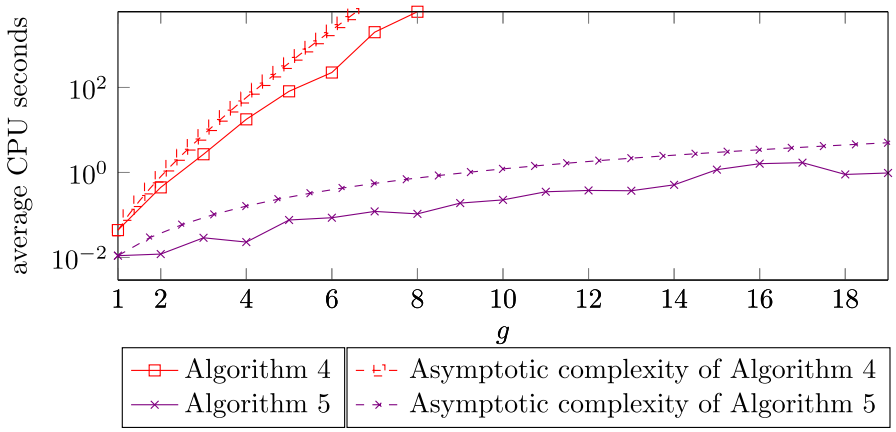

Fig. 2 Empirical timing results and asymptotic complexities for varying g ($n = 2$, $q = 3$, $h_{O_F} = 1$, $\deg c = 1$, irreducible c)

Gaál–Pohst took 0.059 CPU seconds on average. Already for $g = 2$, Gaál–Pohst ran more than 1 day. For $g \geq 9$, Algorithm 4 ran over a day. In all test examples for varying g, as expected, Algorithm 5 was the fastest, and the precomputation took longer than the time required by Algorithm 5 to solve a norm equation.

7 Conclusion

There are several interesting opportunities for future work to extend the results herein. An interesting question is how the quantity $\gcd(q, n)$ affects the performance of the Gaál–Pohst algorithm. Numerical examples for fields $F/\mathbb{F}_q$ with $\gcd(q, n) > 1$ show that this method tends to search a larger space compared to base fields with $\gcd(q, n) = 1$. It is unclear whether this arises from wild ramification; a more careful investigation of the search space and search strategy may lead to efficiency improvements in this setting.

A related problem of interest is to develop algorithms for solving norm equations in lower dimensional submodules M of O_F. A challenge arising in this setting is that we can no longer consider solutions in M up to associates. This is because for an element $\alpha \in M$, an element that is associate to α is not guaranteed to belong to M. In [13], S-unit equations were used to solve this problem.

It would be beneficial to have an algorithm for computing compact representations that does not require F to have an infinite place of degree 1. This restriction, imposed in [9] and necessary for Algorithm 3, guarantees exact equality in (4), rather than the power product just being close to α. A different method is needed to ensure accuracy of compact representations, absent an infinite place of degree 1.

Finally, our results strongly suggest that it would be promising to explore the use of compact representations in solving other families of Diophantine equations over global function fields.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1007/s40993-024-00606-6.

Acknowledgements
The second and third authors' research is supported by NSERC of Canada. We thank our anonymous referees for their constructive and helpful comments.

Data availability The data generated by the experiments described and reported on herein can all be reproduced by the code available on the first author's GitHub [17].

Author details
[1]Department of Mathematics and Statistics, University of Calgary, 2500 University Drive NW, Calgary, AB T2N 1N4, Canada, [2]Department of Computer Science, University of Calgary, 2500 University Drive NW, Calgary, AB T2N 1N4, Canada.

Received: 26 January 2024 Accepted: 9 September 2024
Published online: 21 December 2024

References

1. Alman, J., Vassilevska Williams, V.: A refined laser method and faster matrix multiplication. In: Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 522–539. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (2021)
2. Bauch, J.-D.: Genus computation of global function fields. J. Symb. Comput. **66**, 8–20 (2015)
3. Bauch, J.-D.: Lattices over polynomial rings and applications to function fields. arXiv Preprint arXiv:1601.01361 [math.NT] (2016)
4. Bauch, J.-D., Tran, H., Leem, S.: Fast arithemtic in the divisor class group. Unpublished manuscript (2020)
5. Becker, A., Gama, N., Joux, A.: Solving shortest and closest vector problems: the decomposition approach. IACR Cryptol. ePrint Arch. **2013**, 685 (2013)
6. Birmpilis, S., Labahn, G., Storjohann, A.: Deterministic reduction of integer nonsingular linear system solving to matrix multiplication. In: ISSAC'19—Proceedings of the 2019 ACM International Symposium on Symbolic and Algebraic Computation, pp. 58–65. ACM, New York (2019)
7. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. J. Symb. Comput. **24**(3–4), 235–265 (1997). (**Computational algebra and number theory (London, 1993)**)
8. Diem, C.: Index calculus in class groups of plane curves of small degree. Cryptology ePrint Archive, Paper 2005/119. https://eprint.iacr.org/2005/119 (2005)
9. Eisenträger, K., Hallgren, S.: Computing the unit group, class group, and compact representations in algebraic function fields. In: ANTS X—Proceedings of the Tenth Algorithmic Number Theory Symposium, Volume 1 of Open Book Series, pp. 335–358. Mathematical Sciences Publishers, Berkeley, CA (2013)
10. Fieker, C., von Jurk, A., Pohst, M.E.: On solving relative norm equations in algebraic number fields. Math. Comput. **66**(217), 399–410 (1997)
11. Fincke, U., Pohst, M.E.: Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Math. Comput. **44**(170), 463–471 (1985)
12. Gaál, I., Pohst, M.E.: On solving norm equations in global function fields. J. Math. Cryptol. **3**(3), 237–248 (2009)
13. Gaál, I., Pohst, M.E.: Diophantine equations over global function fields. IV. S-unit equations in several variables with an application to norm form equations. J. Number Theory **130**(3), 493–506 (2010)
14. Harvey, D., van der Hoeven, J.: Polynomial multiplication over finite fields in time $O(n \log n)$. https://hal.archives-ouvertes.fr/hal-02070816/document (2019)
15. Hess, F.: Zur Divisorenklassengruppenberechnung in globalen Funktionenkörpern. PhD Thesis, Technical University Berlin (1999)
16. Labahn, G., Neiger, V., Zhou, W.: Fast, deterministic computation of the Hermite normal form and determinant of a polynomial matrix. J. Complex. **42**, 44–71 (2017)
17. Leem, S.: Solving norm equations over global function fields using compact representation-codes. https://github.com/s-leem/FF_NormEq_CR (2023)
18. Leem, S.: Solving Norm equations over global function fields using compact representations. PhD Thesis, University of Calgary (2023)
19. Lenstra, A.K., Lenstra, H.W., Jr., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. **261**(4), 515–534 (1982)
20. Mahler, K.: Inequalities for ideal bases in algebraic number fields. J. Austral. Math. Soc. **4**, 425–448 (1964)
21. Pohst, M.E., Zassenhaus, H.J.: Algorithmic Algebraic Number Theory. Encyclopedia of Mathematics and Its Applications, vol. 30. Cambridge University Press, Cambridge (1989)
22. Rosen, M.: Number Theory in Function Fields. Graduate Texts in Mathematics, vol. 210. Springer-Verlag, New York (2002)
23. Scheidler, R.: Compact representation in real quadratic congruence function fields. In: Algorithmic Number Theory (Talence, 1996), Volume 1122 of Lecture Notes in Computer Science, pp. 323–336. Springer, Berlin (1996)
24. Scheidler, R.: Decision problems in quadratic function fields of high genus. J. Complex. **16**(2), 411–423 (2000)
25. Siegel, C.L.: Normen algebraischer Zahlen. Nachr. Akad. Wiss. Göttingen Math Phys. Kl. II, pp. 197–215 (1973)
26. Simon, D.: Solving norm equations in relative number fields using *S*-units. Math. Comput. **71**(239), 1287–1305 (2002)
27. Stichtenoth, H.: Algebraic Function Fields and Codes, Volume 254 of Graduate Texts in Mathematics, 2nd edn. Springer-Verlag, Berlin (2009)
28. Tang, A.: Infrastructure of function fields of unit rank one. PhD Thesis, University of Calgary (2011)

29. von zur Gathen, J., Gerhard, J.: Modern Computer Algebra, 3rd edn. Cambridge University Press, Cambridge (2013)

Publisher's Note