ALGORITHMS FOR MULTIVARIATE POLYNOMIALS

DANIEL SCHULTZ

ABSTRACT. Algorithms for multivariate polynomials in flint are discussed. This is a separate compile available outside of the flint2 source tree.

1. Introduction

A polynomial $A \in R[x_1, \dots, x_n]$ is representation as a sums of terms

$$A = t_1 + \cdots + t_a$$

where the terms are ordered as $t_1 > t_2 > \cdots > t_a$ according to some term ordering. The basic operations of addition and subtraction are then equivalent to a merge operation and run in time proportional to the sum of the input term counts.

2. Monomial Representation

The `mpoly` module implements the low level packing and unpacking of exponents for multivariate polynomials. If the variables in the polynomial are, say, x, y and z with $x > y > z$ in the monomial ordering, then the monomial $x^a y^b z^c$ is represented as the array $\{a, b, c\}$ from the user's perspective.

Polynomial exponents are stored in packed format. This means that monomials are actually stored as an array of integer 'fields' that may be packed within a machine word or across multiple machine words if needed. This facilitates basic operations on the monomials, and we make the following assumptions about the correspondence between the variables' exponents and the fields in the packing:

(1) The monomial ordering is a total ordering, i.e. 1 is the smallest.
(2) Multiplication of monomials corresponds to field-wise addition.
(3) Monomials can be compared by comparing their packed representation possibly with an xor mask on certain fields.
(4) The exponent of each variable is itself one of the fields.
(5) The fields are all non-negative.

For the three supported ordering `ORD_LEX`, `ORD_DEGLEX`, and `ORD_DEGREVLEX`, the monomial $x^a y^b z^c$ is converted into fields in the following ways (the least significant field is on the left, the most significant is on the right), and the comparison mask is shown below.

```
ORD_LEX:          | c | b | a |         ( 3 fields)
                   000 000 000

ORD_DEGLEX:       | c | b | a | a+b+c | ( 4 fields)
                   000 000 000  00000
```

```
ORD_DEGREVLEX:    | a | b | c | a+b+c |   ( 4 fields)
                   111 111 111 0000000
```

If one wanted to support, for example, a block ordering which was ORD_DEGLEX in x, y and ORD_DEGREVLEX in z, w with $x > y > z > w$, the monomial $x^a y^b z^c w^d$ would need to be stored as

```
| c | d | c+d | b | a | a+b |     (6 fields)
 111 111 00000 000 000 00000
```

No such interface is currently implemented.

There is no limit to the size of the fields. The fields themselves are packed to a uniform bit width, usually denoted by `bits` in the functions. This bit count should contain an extra sign bit used for overflow detection. Thus, if the maximum field is 15, then the fields only fit into a packing with `bits >= 5`. The total number of machine words taken by an exponent packed into fields is usually denoted by `N` in the code.

If `bits <= FLINT_BITS` then precisely a maximum of `floor(FLINT_BITS/bits)` number of fields may be packed into a single word. Within a word, the packing is from low to high, and unused fields (as well as unused bits) at the top of the word are zero.

3. Multiplication

3.1. Dense multiplication in $\mathbb{Z}[x_1, \dots, x_n]$ or $\mathbb{Z}_p[x_1, \dots, x_n]$.

Given $A(x_1, \dots, x_n), B(x_1, \dots, x_n) \in R[x_1, \dots, x_n]$, set $r_i = 1 + \deg_{x_i}(a) + \deg_{x_i}(b)$. The Kronecker substitution

$$x_1 \to x, \quad x_2 \to x^{r_1}, \quad x_3 \to x^{r_1 r_2}, \quad \dots, \quad x_n \to x^{r_1 \cdots r_{n-1}}$$

gives two univariate polynomials to multiply in $\mathbb{Z}[x]$ or $\mathbb{Z}_p[x]$. This Kronecker substitution is chosen so that it can be reversed to find $A \cdot B \in R[x_1, \dots, x_n]$ from the univariate product. The flint functions `_mpoly_mul_{dense|array}` implement such techniques. The `dense` functions use the ordinary polynomial multiplication functions while the `array` functions use a multiply and accumulate technique that might be better for semi-sparse polynomials.

3.2. Sparse multiplication in $\mathbb{Z}[x_1, \dots, x_n]$ or $\mathbb{Z}_p[x_1, \dots, x_n]$.

Given $A = t_1 + \cdots + t_a, B = s_1 + \cdots + s_b, \in R[x_1, \dots, x_n]$, we need to calculate all products $t_i s_j$, sort them, and combine like terms. This is done using a heap in the functions `_mpoly_mul_johnson` as in [2]. The essential idea is to read off the product terms in order from a heap. The heap never needs to become too large if one uses the relations

$$t_i s_j > t_{i+1} s_j, \quad t_i s_j > t_i s_{j+1}.$$

4. Division

The techniques used for multiplication (Kronecker substitutions in the dense case and heaps in the sparse case) apply to division as well.

5. Powering

Implements a corrected version of an algorithm called FPS in [3]. The basic idea is to map the problem to $R[x]$ via a Kronecker substitution and use a recursion for the coefficients of f^k derived from

$$f(f^k)' = kf'(f^k).$$

Since solving for the coefficients of f^k involves division, this requires some modification for $R = \mathbb{Z}_p$.

6. Interpolation

All of the interpolation methods for $f(x_1, \ldots, x_n) \in R[x_1, \ldots, x_n]$ require strict degree bounds r_i with $\deg_{x_i}(f) < r_i$.

6.1. **Dense Newton Interpolation.** Straightforward, variable-by-variable, recursive, dense interpolation. Number of probes to f is $\prod_i r_i$. There is only one problem with this approach.

- insufficient evaluation points

6.2. **Sparse Zippel Interpolation.** Similar to Newton interpolation, but we use the assumption that monomials don't disappear under evaluation. For example, suppose r_x, r_y, r_z are the strict degree bounds. We first find $f(x, 1, 1)$ using dense interpolation with r_x values of x, say $x_1, \ldots, x_{r_x},$. If

$$f(x, 1, 1) = x^5 + 2x^2 + 1,$$

we make the assumption that

$$f(x, y, 1) = a_1(y)x^5 + a_2(y)x^2 + a_3(y),$$

and proceed to interpolate the $a_i(y)$ using dense univariate interpolation in y. We need r_y values of y, say $y_1, \ldots, y_{r_y}$. For each of these values $y = y_i$ we can find the coefficients (?) in

$$f(x, y_i, 1) = (?)x^5 + (?)x^2 + (?)$$

by plugging in *three* random values of x and solving the linear system. To find $f(x, y, 1)$ at this point the number of probes to f we have used is $r_x + 3r_y$, which is probably fewer than $r_x r_y$.

Now suppose we obtain

$$f(x, y, 1) = y^2x^5 + x^2 + y^7x^2 + y^3.$$

Make the assumption

$$f(x, y, z) = b_1(z)y^2x^5 + b_2(z)x^2 + b_3(z)y^7x^2 + b_4(z)y^3,$$

and interpolate the $b_i(z)$ using dense univariate interpolation in z. We need r_z values of z, say $z_1, \ldots, z_{r_z}$. For each of these values $z = z_i$ we can find the coefficients (?) in

$$f(x, y, z) = (?)y^2x^5 + (?)x^2 + (?)y^7x^2 + (?)y^3$$

by plugging in *four* random pairs of values of (x, y) and solving the linear system. To find $f(x, y, z)$ at this point the number of probes to f we have used is $r_x + 3r_y + 4r_z$, which is probably fewer than $r_x r_y r_z$.

This approach has an additional problems.

- insufficient evaluation points
- inconsistent/underdetermined linear equations
- associated linear algebra costs

6.3. **Sparse Interpolation with the Berlekamp-Massey Algorithm.** Given the strict degree bounds r_i, in order to interpolate $f(x_1, \dots, x_n)$ it suffices to interpolate $f(\xi, \xi^{r_1}, \xi^{r_1 r_2}, \dots, \xi^{r_1 \cdots r_{n-1}})$, which is a univarate with degree bound $\prod_i r_i$. If t is the number of terms of f, then we can summarize the probe counts of the three methods.

1. dense: $\prod_i r_i$
2. zippel: approximately $t \cdot \sum_i r_i$.
3. bma: $2t$.

This approach has problems too.

- insufficient evaluation points
- costs of the associated linear algebra and discrete logarithms.

Since the presentation in [7] is overly complicated and does not deal with the half gcd, it seems reasonable to review the Berlekamp-Massey Algorithm here. Given a formal power series

$$\frac{a_1}{x} + \frac{a_2}{x^2} + \frac{a_3}{x^3} + \cdots, \quad a_i \in \mathbb{F}$$

vanishing at $x = \infty$ and the fact that this power series represents a rational function, we are interested in computing this rational function. The following theorem says that we can use the extended euclidean algorithm and stop when the first remainder of degree $< \frac{n}{2}$ is obtained.

Theorem 6.1. *Suppose that*

$$\frac{a_1}{x} + \frac{a_2}{x^2} + \frac{a_3}{x^3} + \cdots = -\frac{\bar{u}}{\bar{v}}$$

for some $\bar{u}, \bar{v} \in \mathbb{F}[x]$ *with* $\deg(\bar{u}) < \deg(\bar{v}) \le \frac{n}{2}$. *Suppose further that*

$$u x^n + v(a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_{n-1} x + a_n) = r \tag{6.1}$$

for some $u, v, r \in \mathbb{F}[x]$ *with* $\deg(u) < \deg(v) \le \frac{n}{2}$ *and* $\deg(r) < \frac{n}{2}$ *and* $\deg(r) < \deg(v)$. *Then,*

$$\frac{\bar{u}}{\bar{v}} = \frac{u}{v}.$$

Proof. Dividing both sides of (6.1) by vx^n shows that

$$\frac{\bar{u}}{\bar{v}} = \frac{u}{v} + O\left(\frac{1}{x^{n+1}}\right),$$

which, on account of the degree bounds $\deg(\bar{v}), \deg(v) \le \frac{n}{2}$, proves the equality. □

This reconstruction may be applied to reconstruct an $f(\xi) = c_1 \xi^{e_1} + \cdots + c_t \xi^{e_t} \in \mathbb{F}[\xi]$ from the sequence of evaluation points

$$a_i = f(\alpha^{s+i-1}), \quad \alpha \ne 0, \quad s \in \mathbb{Z},$$

for in this case we have

$$\frac{a_1}{x} + \frac{a_2}{x^2} + \frac{a_3}{x^3} + \cdots = \frac{c_1 \alpha^{e_1 s}}{x - \alpha^{e_1}} + \cdots + \frac{c_t \alpha^{e_t s}}{x - \alpha^{e_t}}.$$

If this rational function is known and the e_i can be found, then f is known as well.

The main problem with this approach is that the term bound t is not known in advance. The approach we take is to calculate the v in (6.1) for some n points $a_1, \dots, a_n$. Then, we add another m points to form the sequence $a_1, \dots, a_{n+m}$ and calculate the corresponding v'. If $v = v'$, then it is likely that v is the correct denominator. The extent to which previous computations may be reused is addressed in Theorem 6.3. We follow [8] for the presentation of the half gcd. An elementary matrix is one of the form $\left(\begin{smallmatrix} 0 & 1 \\ 1 & q \end{smallmatrix}\right)$ for $\deg(q) > 0$ and a regular matrix is a product of zero or more elementary matrices. The notation $U \xrightarrow{M} V$ shall mean that M is a regular matrix and $U = MV$. If $\deg(A) > \deg(B)$ then $\operatorname{hgcd}(A, B)$ is defined (see [8]) as the (unique) regular matrix M such that

$$\begin{pmatrix} A \\ B \end{pmatrix} \xrightarrow{M} \begin{pmatrix} C' \\ D' \end{pmatrix},$$

$$\deg(C') \geq \frac{\deg(A)}{2} > \deg(D').$$

Theorem 6.2. *Suppose that*

$$\begin{pmatrix} A_0 \\ B_0 \end{pmatrix} \xrightarrow{M} \begin{pmatrix} A_0' \\ B_0' \end{pmatrix}$$
$$\deg(A_0') > \deg(B_0')$$
$$\deg(A_0) \leq 2\deg(A_0')$$

Then, for any A_1, B_1 *with* $\deg(A_1), \deg(B_1) < m$,

$$\begin{pmatrix} A_0x^m + A_1 \\ B_0x^m + B_1 \end{pmatrix} \xrightarrow{M} \begin{pmatrix} A' \\ B' \end{pmatrix}$$
$$\deg(A') = m + \deg(A_0')$$
$$\deg(B') \leq m + \max(\deg(B_0'), \deg(A_0') - 1)$$
$$\deg(B') \leq m + \max(\deg(B_0'), \frac{\deg(A_0)}{2} - 1)$$

for some A', B'.

Proof. This is a trivial rearrangement of Lemma 1 in [8]. ☐

Theorem 6.3. *Suppose* $\deg(s_n) < n$, $\deg(s_m) < m$ *and*

$$\begin{pmatrix} x^n \\ s_n \end{pmatrix} \xrightarrow{M} \begin{pmatrix} r_0 \\ r_1 \end{pmatrix}$$
$$\deg(r_0) \geq \frac{n}{2} > \deg(r_1)$$

Then, a regular matrix M' *(and thus* r_0', r_1'*) such that*

$$\begin{pmatrix} x^{n+m} \\ s_nx^m + s_m \end{pmatrix} \xrightarrow{M'} \begin{pmatrix} r_0' \\ r_1' \end{pmatrix}$$
$$\deg(r_0') \geq \frac{n+m}{2} > \deg(r_1')$$

may be calculated as follows. Define A', B' *by*

$$\begin{pmatrix} x^{n+m} \\ s_nx^m + s_m \end{pmatrix} \xrightarrow{M} \begin{pmatrix} A' \\ B' \end{pmatrix}$$

It will be the case that $\deg(A') \geq \frac{n+m}{2}$. *If* $\frac{n+m}{2} > \deg(B')$, *return with* $M' = M$. *Otherwise set* $C = B'$, $D = \mathrm{rem}(A', B')$ *and* $q = \mathrm{quo}(A', B')$. *Define* $k := n + m - \deg(C)$. *It will be the case that* $0 < k \leq \deg(C)$. *Return with*

$$M' = M \cdot \begin{pmatrix} 0 & 1 \\ 1 & q \end{pmatrix} \cdot \mathrm{hgcd} \begin{pmatrix} \mathrm{quo}(C, x^k) \\ \mathrm{quo}(D, x^k) \end{pmatrix}$$

Proof. By Theorem 6.2, $\deg(A') = m + \deg(r_0) \geq m + \frac{n}{2} \geq \frac{n+m}{2}$. Now suppose $\frac{n+m}{2} \leq \deg(B')$, from which the assertion $k \leq \deg(C)$ follows automatically. By Theorem 6.2, $\deg(B') \leq m + \max(\deg(r_1), \deg(r_0) - 1) < m + n$. Thus, the assertion $0 < k$ is proved. Finally, suppose

$$\begin{pmatrix} C_0 := \mathrm{quo}(C, x^k) \\ D_0 := \mathrm{quo}(D, x^k) \end{pmatrix} \xrightarrow{H} \begin{pmatrix} C_0' \\ D_0' \end{pmatrix},$$

$$\deg(C_0') \geq \frac{\deg(C_0)}{2} > \deg(D_0').$$

If C', D' are defined by

$$\begin{pmatrix} C \\ D \end{pmatrix} \xrightarrow{H} \begin{pmatrix} C' \\ D' \end{pmatrix},$$

it suffices to prove that $\deg(C') \geq \frac{n+m}{2} > \deg(D')$. By Theorem 6.2,

$$\begin{aligned} \deg(C') &= k + \deg(C_0') \\ &\geq k + \frac{\deg(C_0)}{2} \\ &= k + \frac{\deg(C) - k}{2} \\ &= \frac{n+m}{2}. \end{aligned}$$

Also by Theorem 6.2,

$$\begin{aligned} \deg(D') &\leq k + \max(\deg(D_0'), \frac{\deg(C_0)}{2} - 1) \\ &< k + \max(\frac{\deg(C_0)}{2}, \frac{\deg(C_0)}{2}) \\ &= \frac{n+m}{2}. \end{aligned}$$

□

7. Greatest Common Divisor

7.1. **Dense GCD in $\mathbb{Z}_p[x_1, \ldots, x_n]$.** Brown's algorithm [1] is used here. This comes in two versions - a small prime version and a large prime version. These refer not to the size of the p's involved, but rather to the field from which evaluation points are chosen: it can either be $\mathbb{F}_p$ or an extension of $\mathbb{F}_p$. The small prime version interpolates in each variable by choosing evaluation points from $\mathbb{F}_p$. If this fails, then the large prime method uses interpolation in $\mathbb{F}_p/(f(x_n))[x_1, \ldots, x_{n-1}]$, i.e. $\mathbb{F}_q[x_1, \ldots, x_{n-1}]$, for sufficiently many irreducible $f(x) \in \mathbb{Z}_p[x]$. No explicit divisibility checks need to be performed because the cofactors are reconstructed along with the GCD.

7.2. **Dense GCD in** $\mathbb{Z}[x_1,\ldots,x_n]$**.** We simply reconstruct the GCD from its image in $\mathbb{Z}_p[x_1,\ldots,x_n]$ for sufficiently many p. Only large p's are used, and dense GCD's in $\mathbb{Z}_p[x_1,\ldots,x_n]$ only use the small prime version. Each image GCD in $\mathbb{Z}_p$ is correct and Brown's coefficient bounds [1] are used instead of a divisibility check. Some pseudocode is Section 10.

7.3. **Sparse GCD in** $R[x_1,\ldots,x_n]$**.** Assuming that we have a gcd algorithm for $R[x_1,\ldots,x_m]$, we can view the inputs as elements of $R[x_1,\ldots,x_m][x_{m+1},\ldots,x_n]$ and use interpolation to extend this algorithm from m variables to n variables. Brown's algorithm corresponds to taking $m=n-1$, using univariate interpolation for the extension of $n-1$ variables to n variables, and recursively solving the $n-1$ variable gcd problem with the Euclidean algorithm as the base case. Taking $m=1$ gives Zippel's approach [4]. If the inputs are made primitive with respect to $x_1,\ldots,x_m$ by factoring out polynomials in $R[x_{m+1},\ldots,x_n]$, the gcd of the leading coefficients of the input with respect to $x_1,\ldots,x_m$ may be imposed as the leading coefficient of the interpolated gcd. Finally, if the primitive part with respect to $x_1,\ldots,x_m$ of this interpolated gcd divides both inputs, it must be the true gcd.

Of note here is an algorithm stated slightly incorrectly in [6] and [5]; The basic idea is to reconstruct the correct leading term of the gcd $\in R[x_1,\ldots,x_m]$ using some linear algebra directly instead of constructing some known multiple and then removing content. This is in `fmpz_mpolyl_gcd_zippel` and a rough overview is:

```
fmpz_mpolyl_gcdm_zippel($A, B \in \mathbb{Z}_p[x_1,\ldots,x_n][X]$, $n \geq 1$):
    The GCD is assumed to have no content w.r.t. $X$ (content in $\mathbb{Z}[x_1,\ldots,x_n]$)
    Pick a prime $p$ and call nmod_polyl_gcdp_zippel to get an probable image of $G \mod p$
    Assume that the true gcd $G$ over $\mathbb{Z}$ has the same monomials as this image mod $p$.
    Pick more primes $p$ and call nmod_mpolyl_gcds_zippel to get more images of $G \mod p$.
    Combine the images via chinese remaindering and test divisibility.
```

The "p" versions produce a correct gcd when the inputs have no content in $\mathbb{F}_p[x_1,\ldots,x_n]$.

```
nmod_mpolyl_gcdp_zippel($A, B \in \mathbb{F}_p[x_1,\ldots,x_n][X]$, $n \geq 1$):
    If the GCD has content w.r.t. $X, x_1, \ldots, x_1$ (content in $\mathbb{F}_p[x_n]$), fail.
    Pick an evaluation point $x_n \to \alpha$ for $\alpha \in \mathbb{F}_p$.
    (1) Call nmod_mpolyl_gcdp_zippel recursively on the evaluated inputs in $\mathbb{F}_p[x_1,\ldots,x_{n-1}][X]$.
    Record the form $f$ of the GCD obtained for step (2) below.
    Pick severial evaluation points $x_n \to \alpha$ for $\alpha \in \mathbb{F}_q$.
    (2) Call [fq_]nmod_mpoly_gcds_zippel on the evaluated inputs in $\mathbb{F}_q[x_1,\ldots,x_{n-1}][X]$.
    Combine the answer from (1) and the answers from (2) via interpolation in $x_n$.
    Check divisibility on the proposed interpolated GCD.
```

The "s" versions are the heart of Zippel's sparse interpolation.

```
nmod_mpolyl_gcds_zippel($A, B \in \mathbb{F}_q[x_1,\ldots,x_n][X]$, assumed monomial form $f$ of gcd):
    Via evaluations of the form $(x_1,\ldots,x_n) \to (\alpha_1,\ldots,\alpha_n) \in \mathbb{F}_p^n$,
    and GCD computations in $\mathbb{F}_p[X]$, and linear algebra, try to compute the coefficients
    of the assumed form $f$ to match the GCD of the inputs (up to scalar multiples in $\mathbb{F}_p$).
```

7.4. **PRS.** The PRS algorithm works over any gcd domain R. It starts with a primitive input with respect to some main variable and calculates a pseudo gcd with a pseudo remainder sequence. Content is removed from the pseudo gcd to produce the true gcd by a recursive call. The final content can be computed without expensive recursive calls to gcd in the case when we know the leading or trailing coefficient in the main variable must be a monomial in the remaining variables.

This algorithm has been discarded because it is so bad but may be reintroduced for low degrees.

7.5. **Hensel Lifting.** The gcd can also be calculated using Hensel lifting [10]. The gcd of the resulting univariates when all variables but one are substituted away gives two factorizations which can be lifted to obtain the multivariate gcd.

8. Factorization

8.1. **Squarefree Factorization in $K[x_1, \dots, x_n]$.** By taking derivatives and greatest common divisors, we may assume that the input polynomial is squarefree and primitive with respect to each variable. Thus in characteristic zero the input polynomial $f \in K[x_1, \dots, x_n]$ may be assumed to satisfy

$$\forall_i \quad f_{x_i} \neq 0 \quad \text{and} \quad \gcd(f, f_{x_i}) = 1.$$

Over a finite field ($K = \mathbb{F}_q$) of characteristic p, we have the slightly weaker conditions

$$\begin{aligned} & f_{x_1} \neq 0 \quad \text{and} \quad \gcd(f, f_{x_1}) = 1 \\ \forall_{i>1} \quad & f_{x_i} = 0 \quad \text{or} \quad \gcd(f, f_{x_i}) = 1 \end{aligned} \tag{8.1}$$

While we could apply the factorization algorithms directly to this f with x_1 as the main variable, it is possible to a bit better when some of the other derivatives vanish.

Theorem 8.1. *With the assumption 8.1 on f and prime powers $p^{e_2}, \dots, p^{e_n}$ and a deflated polynomial g with*

$$g(x_1, x_2^{p^{e_2}}, \dots, x_n^{p^{e_n}}) = f(x_1, x_2, \dots, x_n),$$

the factorization of f is the inflated factorization of g.

The proof follows by induction from the following lemma: the polynomials $g(x_1, x_2, x_3, \dots, x_n)$ and $g(x_1, x_2^p, x_3, \dots, x_n)$ have the same factorization (up to inflation $x_2 \to x_2^p$).

Lemma 8.2. *If $p = \operatorname{char}(K) > 0$ and $f(x, y) \in K[x, y] \setminus (K[x] \cup K[y])$ is irreducible and $f(x^p, y)$ is squarefree, then $f(x^p, y)$ is irreducible.*

Proof. Suppose that $f(x^p, y) = g(x, y)h(x, y)$ for $g, h \notin K$. Since $f(x^p, y)$ is squarefree, g and h are squarefree, and there are $s, t \in K(y)[x]$ with $1 = sg+th$. By differentiating $f(x^p, y) = g(x, y)h(x, y)$, we obtain $0 = hg_x+gh_x$, which when combined with $1 = sg+th$ gives $h(th_x-sg_x) = h_x$. This implies that $h_x = 0$ and in turn that $g_x = 0$, which implies that $f(x, y)$ is reducible, a contradiction. □

8.2. **Factorization in $R[x]$.**

8.2.1. *Quadratic over characteristic $\neq 2$.* The primitive polynomial $ax^2 + bx + c$ factors if and only if $b^2 - 4ac$ is a square in R, in which case the factors are the primitive parts of $2ax + b \pm \sqrt{b^2 - 4ac}$.

8.2.2. *Quadratic in $R[X]$ for $R = \mathbb{F}_{2^k}[x_1, \dots, x_n]$.* We wish to determine if $X^2 + AX + B$ has a root in R. Since $X_0 + A$ is a root if X_0 is, at least one of the two roots does not have $\operatorname{lt}(A)$ as a term. (It very well may be the case that both roots have a monomial matching $\operatorname{lm}(A)$, but then both corresponding coefficients must be different from the leading coffcient of A). Therefore, we make the important assumption that *we are searching for a root X_0 with $\operatorname{lt}(A)$ not a term of X_0.* Let m denote the leading term of X_0. By taking leading terms in $X_0^2 + AX_0 + B$ and applying the assumption, we have

$$\operatorname{lt}(m^2 + \operatorname{lt}(A)m) = \operatorname{lt}(B), \quad \text{and} \quad m \neq \operatorname{lt}(A).$$

For any specific given terms $\operatorname{lt}(A)$, $\operatorname{lt}(B)$, this equation is easy to solve for m or to determine that there is no solution.

$$\begin{aligned}
\operatorname{lm}(m) > \operatorname{lm}(A) : &\quad m = \sqrt{\operatorname{lt}(B)} \\
\operatorname{lm}(m) = \operatorname{lm}(A) : &\quad m = \zeta / \operatorname{lc}(A) \sqrt{\operatorname{lm}(B)}, \quad \zeta^2 + \zeta = \operatorname{lc}(B) / \operatorname{lc}(A)^2 \\
\operatorname{lm}(m) < \operatorname{lm}(A) : &\quad m = \operatorname{lt}(B) / \operatorname{lt}(A)
\end{aligned}$$

Once m is found, the equation satisfied by $X_0 - m$ has the same A and a new B with a smaller leading monomial. In this way the solution may be written down in order, and this process is a simplification of Sections 4 and 5 in [14], which does not present a sparse algorithm due to the many (possibly disastrous) divisions performed. The quadratic $\zeta^2 + \zeta + c \in \mathbb{F}_{2^k}$ has a root if and only if $\operatorname{Tr}(c) = 0$, in which case $c = \sum_{i=1}^{k-1} c^{2^i} \sum_{j=0}^{i-1} u^{2^j}$ is a root where u is any element of $\mathbb{F}_{2^k}$ with $\operatorname{Tr}(u) = 1$. If $\mathbb{F}_{2^k} = \mathbb{F}_2[\theta]/P(\theta)$, then $u = 1/(\theta P'(\theta))$ will do.

8.2.3. *Cubic over $\mathbb{Z}$.* To factor a cubic over $\mathbb{Z}$, we first find the roots over the more friendly ring $\mathbb{Z}_2$ and then test these roots over $\mathbb{Z}$. Since it is easy to bound the roots over $\mathbb{Z}$, the roots over $\mathbb{Z}_2$ only need to be calculated to some finite precision p, that is, to order $O(2^p)$.

Factor $x^3 + 2^\alpha ax + 2^\beta b$ over $\mathbb{Z}_2$ where $\alpha, \beta \geq 0$ and a, b are odd integers:

(1) $2\beta = 3\alpha$: irreducible, as replacing $x \leftarrow 2^{\beta/3}y$ has no roots modulo 2 for y.
(2) $2\beta < 3\alpha$:
   (a) $3 \nmid \beta$: irreducible as all roots have valuation $\beta/3$.
   (b) $3 \mid \beta$: Replacing $x \leftarrow 2^{\beta/3}y$ gives $y^3 + 2^{\alpha - 2\beta/3}ay + b = 0$, which factors as $(y^2 + y + 1)(y+1) = 0$ modulo 2. Hence there is a unique root in $\mathbb{Z}_2$, and this root has valuation $\beta/3$.
(3) $2\beta > 3\alpha$: Replacing $x \leftarrow 2^{\beta-\alpha}y$ gives $2^{2\beta-3\alpha}y^3 + ay + b = 0$, which has $y = 1 \bmod 2$ as a root. This gives a factorization

$$2^{2\beta-3\alpha}y^3 + ay + b = (y + r)(2^{2\beta-3\alpha}y^2 - 2^{2\beta-3\alpha}ry + s)$$

for some odd $r, s \in \mathbb{Z}_2$. This becomes

$$(x + 2^{\beta-\alpha}r)(x^2 - 2^{\beta-\alpha}rx + 2^\alpha s) = 0$$

   (a) $2 \nmid \alpha$: quadratic is irreducible and $-2^{\beta-\alpha}r$ is the only root.
   (b) $2 \mid \alpha$: assuming the square roots exist, the roots of the quadratic, which have valuation $\alpha/2$, are

$$2^{\beta-\alpha-1}r \pm 2^{\alpha/2}\sqrt{2^{2\beta-3\alpha-2}r^2 - s}$$

If r and s are calculated to some absolute precision $O(2^p)$, then this expression is also known to absolute precision $O(2^p)$ except when $\alpha = 0$ and $\beta = 1$, in which case the square root loses more than one bit of precision.

8.3. **Factorization in $K[x, y]$.** For $K = \mathbb{Q}$, an irreducible bivariate polynomial $f(x, y)$ remains irreducible modulo $y = y_0$ for a generic $y_0 \in \mathbb{Q}$. Hence, all of the difficult recombination may be pushed to the univariate factorization. When $K = \mathbb{F}_q$ the recombination in [9] is necessary.

8.3.1. *Bivariate factorization over* $\mathbb{Q}$. We begin with $f(x, y)$ satisfying

(1) $f(x, y) \in \mathbb{Z}[x, y]$ and $f(x, 0) \in \mathbb{Z}[x]$ are squarefree so that we can lift.
(2) $f(x, y)$ is primitive with respect to x (i.e. $\operatorname{cont}_x(f) \in \mathbb{Z}[y]$ is 1) so that any factor is also primitive with respect to x.
(3) $\deg_x(f(x, y)) = \deg_x(f(x, 0))$ (i.e. $\operatorname{lc}_x(f)$ does not vanish at $y = 0$) so that we can make f monic.

Let

$$\tilde{f}(x, y) = f(x, y)/\operatorname{lc}_x(f(x, y)) \in \mathbb{Q}[[y]][x]$$

be the monic version of f computed to precision $O(y^{1+\deg_y f})$. We can factor $\tilde{f}(x, 0) \in \mathbb{Q}[x]$ by a univariate algorithm and lift the factors to produce an irreducible factorization in $\mathbb{Q}[[y]][x]$ as

$$\tilde{f}(x, y) = \prod_{i=1}^{l} \tilde{f}_i(x, y).$$

The $\tilde{f}_i(x, y)$ are also monic and need to be computed to precision $O(y^{1+\deg_y f})$. For each subset S of $\{1, \ldots, l\}$, we then have the candidate true factor

$$\operatorname{ppart}_x\left(\operatorname{lc}_x(f) \prod_{i \in S} \tilde{f}_i(x, y)\right),$$

where, before taking the primitive part, the elements of $\mathbb{Q}[[y]][x]$ must be mapped to $\mathbb{Q}[y][x]$ via remainder upon division by $y^{1+\deg_y f}$. Since we are only interested in candidate factors over $\mathbb{Z}$, $\mathbb{Q}$ may be replaced by $\mathbb{Z}/p^k\mathbb{Z}$ for appropriate p^k (in particular $p \nmid \operatorname{lc}_x(f(x, 0))$). The coefficients in $\mathbb{Z}/p^k\mathbb{Z}$ must then be mapped to $\mathbb{Z}$ via the symmetric remainder before taking the primitive part.

8.3.2. *Bivariate Factorization over* $\mathbb{F}_q$. We begin with $f(x, y) \in \mathbb{F}_q[x, y]$ and an irreducible $\alpha(y) \in \mathbb{F}_q[y]$ (with $\mathbb{F}_{q^k} := \mathbb{F}_q[y]/\alpha(y)$) such that

(1) $\alpha(y)$ does not divide $\operatorname{lc}_x f(x, y)$ so that we can make f monic.
(2) $f(x, y) \bmod \alpha(y) \in \mathbb{F}_{q^k}[x]$ is squarefree so that we can lift.
(3) $f(x, y)$ is primitive with respect to x.

The irreducible factorization of $f(x, y) \bmod \alpha(y)$ can be lifted to a monic factorization in $\mathbb{F}_q[[\alpha(y)]][x]$. With the help of some linear algebra over $\mathbb{F}_p$ these factors can be recombined into true factors.

8.4. **Factorization in** $R[x_1,\ldots,x_n][X]$**.** Factoring of a multivariate squarefree primitive polynomial f over $R[x_1,...,x_n][X]$ (satisfying the assumptions of Section 8.1 works by reducing f modulo the ideal

$$\langle x_1=\alpha_1, x_2=\alpha_2,\ldots,x_n=\alpha_n\rangle$$

for some $\alpha_i \in R$, factoring the resulting univariate into, say, r, factors, and then lifting the univariate factorization to a multivariate factorization. The evaluation points must be good in the sense that $f(\alpha_1,\ldots,\alpha_n,X)$ is squarefree and has the same degree as $f(x_1,\ldots,x_n,X)$ in X. This lifting process does not change the leading coefficients in X, hence it is necessary that the leading coefficients be "correct" before the lifting. In the most general setting, we can determine $d_i \in R[x_1,...,x_n]$, such that it is known that d_i divides the leading coefficient of the i-th lifted factor. Then, before lifting, we compute $m = \mathrm{lc}_X(f)/(d_1\cdots d_r)$, impose a leading coefficient of $d_i m$ on the i-th factor, and multiply f by m^{r-1}. If the lifting succeeds, then the actual factors can be obtained by taking principle parts. Doing no work to precompute leading coefficients corresponds to taking $d_i = 1$, which can obviously lead to large swells.

8.4.1. *Wang's leading coefficient computation.* Wang [11] has a good solution to the leading coefficient problem over $\mathbb{Z}$. The idea can be illustrated by a simple example.

$$(2x_1^3x_2+2x_1^3x_2)X^2+\cdots=(2x_1(x_1+x_2)X+x_1)(x_1x_2X+6)$$

First the irreducible factorization of the leading coefficient is computed

$$(2x_1^2x_2+2x_1^2x_2)=2x_1^2x_2(x_1+x_2)$$

Next, an evaluation point $x_i=\alpha_i$ such that there exists primes p_i such that

$$\begin{aligned}&p_3 \mid \alpha_1+\alpha_2, \quad p_3\nmid\alpha_2, \quad p_3\nmid\alpha_1,\\ &p_2\mid\alpha_2, \quad p_2\nmid\alpha_1,\\ &p_1\mid\alpha_1\end{aligned}$$

Lets take $\alpha_1=10,\alpha_2=14$ and $p_1=5,p_2=7,p_3=3$. The univariate factorization comes out as

$$20(48X+1)(70X+3)$$

What is of interest here is the leading coefficients of the primitive factors over $\mathbb{Z}$. From $p_3=3$ we can correctly distribute x_1+x_2 to the first multivariate factor. From $p_2=7$ we can distribute x_2^2 to both factors, and from $p_1=5$, we can distribute x_1 to the second factor.

When R is a finite field, there is no useful notion of "prime". Furthermore, the probability that an irreducible univariate factorization can be lifted to a multivariate factorization is low and sometimes zero. Hence this does not work as stated. One may replace R by $R[Y]$ for an auxiliary indeterminate Y and consider polynomial substitutions of the form

$$\begin{aligned}x_1&=\alpha_1+\beta_1Y+\gamma_1Y^2+\cdots\\ x_2&=\alpha_2+\beta_2Y+\gamma_2Y^2+\cdots\\ &\cdots\\ x_n&=\alpha_n+\beta_nY+\gamma_nY^2+\cdots.\end{aligned}$$

The base case factorization is now not $R[X]$ but $R[Y][X]$. The points $\alpha_1,...,\alpha_n$ still need to be good because the lifting will ultimately begin with univariates. However, the univariate factors come not from an irreducible univariate factorization, but from the $Y = 0$ image of a bivariate factorization, which should greatly increases the changes of success in the lifting.

8.4.2. *Kaltofen's leading coefficient computation.* In this recursive approach [15], after substituting away all but *two* of the variables, the bivariate polynomial is factored and the leading coefficients of the bivariate factors can be lifted against the leading cofficient of the original polynomial. Since only squarefree lifting is implemented, it is actually the squarefree parts of everything that are lifted.

8.4.3. *Dense Hensel lifting.* Some pseudocode is Section 10. Of note here is that when lifting over $\mathbb{Z}$, we do not lift over $\mathbb{Z}/p^k\mathbb{Z}$ as Wang [11] advises but do the lifting directly over $\mathbb{Z}$.

8.4.4. *Sparse Hensel lifting.*

9. Absolute Factorization

The goal of absolute factorization is to take an irreducible $f \in R[x_1, \dots, x_n]$ and either determine that f is absolutely irreducible or provide a factorization

$$f = gh, \quad g, h \in R'[x_1, \dots, x_n]$$

where g is absolutely irreducible. h may or may not be absolutely irreducible: it is simply the product of the rest.

9.1. **Absolute Irreduciblity Testing.**

Here we follow Gao [12]. For a multivariate polynomial $f = \sum_{\mathbf{i} \in \mathbb{Z}^n} c_{\mathbf{i}} \boldsymbol{x}^{\mathbf{i}}$, the Newton polygon $N(f)$ is defined to be the convex hull of $\{\mathbf{i} \in \mathbb{Z}^n | c_{\mathbf{i}} \neq 0\}$ in $\mathbb{R}^n$. Since $N(fg) = N(f) + N(g)$ where $+$ denotes the Minkowski sum, if $N(f)$ is indecomposable, then f is absolutely irreducible. Although indecomposability testing is hard, Gao gives a reasonable algorithm in two dimensions [13], that is, for bivariate polynomials, and projects higher dimensional polytopes onto a two-dimensional "shadow" to test them for indecomposability.

If $f \in K[\boldsymbol{x}]$ happens to be irreducible over K but not over the algebraic closure $\overline{K}$, then $N(f)$ will never be sufficient to prove the irreducibility over K. In the case that we are able to prove that f is irreducible over K using other methods, $N(f)$ can still be used to obtain some information on the degree of an extension of K needed to factor f absolutely. An absolute factorization of an irreducible $f(\boldsymbol{x}) \in K[\boldsymbol{x}]$ looks like

$$f(\boldsymbol{x}) = \operatorname{resultant}_\alpha(u(\alpha), g(\alpha, \boldsymbol{x}))$$

for some irreducible $u(\alpha) \in K[\alpha]$ of degree, say, m. Since all m of the $g(\alpha, \boldsymbol{x})$ have the same Newton polygon, it follows that $N(f) = m \cdot N(g)$, and thus m divides the coordinates of every vertex in $N(f)$. This can severely limit the possibilities for the extension degree required for an absolute factorization.

9.2. **Bivariate Absolute Factorization over $\mathbb{Q}$.**

The idea here is that an absolutely irreducible $g(x, y) \in \overline{\mathbb{Q}}[x, y]$ remains absolutely irreducible in $\overline{\mathbb{F}}_p[x, y]$ for generic p.

Assume that $f(x, y) \in \mathbb{Q}[y][x]$ is irreducible. Pick a good $\alpha \in \mathbb{Q}$ and a good rational prime p. The definition of "good" is that none of the following steps or assumptions fail. Determine an $\mathbb{F}_q = \mathbb{F}_{p^?}$ such that $f(x, \alpha)$ splits completely into distinct linear irreducibles:

$$\frac{f(x, \alpha)}{\operatorname{lc}_x(f(x, y))|_{y=\alpha}} = \prod_i x - r_i \text{ in } \mathbb{F}_q[x].$$

Lift this to power series:

$$\frac{f(x,y)}{\mathrm{lc}_x(f(x,y))} = \prod_i x - r_i(y) \text{ in } \mathbb{F}_q[[y-\alpha]][x].$$

Do some linear algebra to recombine the factors into a real factorization:

$$f(x,y) = \prod_j g_j(x,y) \text{ in } \mathbb{F}_q[y][x].$$

The $g_i(x,y) \in \mathbb{F}_q[y][x]$ are absolutely irreducible. It might be possible to reduce the size of q at this point.

We then try to lift this to a factorization in $\mathbb{Q}_q[y][x]$:

$$f(x,y) = \prod_j \widetilde{g}_j(x,y) \text{ in } \mathbb{Q}_q[y][x].$$

In order to attemp this lift the $\mathrm{lc}_x(\widetilde{g}_j(x,y)) \in \mathbb{Q}_q[y]$ must be correct before starting. Assume $\mathrm{lc}_x(f(x,y))$ is monic in y, and that its squarefree part remains squarefree modulo p. Then, the squarefree factors of the $\mathrm{lc}_x\, \widetilde{g}_j(x,y)$ can be lifted and we can recover the monic $\mathrm{lc}_x(\widetilde{g}_j(x,y)) \in \mathbb{Q}_q[y]$.

Finally, we map $\widetilde{g}_1(x,y)$ to some number field $K[x,y]$ (so that the other $\widetilde{g}_j(x,y)$ are its conjugates) and test divisibility $g_1|f$.

9.3. **Bivariate Absolute Factorization over $\mathbb{F}_q$.**

9.4. **Multivariate Absolute Factorization.**

For absolutely factoring an irreducible in $R[x_1,\ldots,x_n][X]$, the plan is to substitute good auxiliary polynomials

$$\begin{aligned} x_1 &= \alpha_1 + \beta_1 Y + \gamma_1 Y^2 + \cdots \\ x_2 &= \alpha_2 + \beta_2 Y + \gamma_2 Y^2 + \cdots \\ &\cdots \\ x_n &= \alpha_n + \beta_n Y + \gamma_n Y^2 + \cdots, \end{aligned}$$

and absolutely factor the resulting bivariate in $R[X,Y]$, and then lift the $Y = 0$ images of the two factors back to a multivariate factorization. This would require the fact that an absolutely irreducible multivariate remains an absolutely irreducible bivariate under a generic substitution of this form.

References

[1] W. S. Brown. On Euclid's Algorithm and theComputation of Polynomial Greatest Common Divisors. J. ACM 18 (1971), 478-504.

[2] Johnson, S.C., 1974. Sparse polynomial arithmetic. ACM SIGSAM Bulletin 8 (3), pp. 63–71.

[3] Monagan M., Pearce R.: Sparse polynomial powering using heaps. "Computer Algebra in Scientific Computing", Springer, 2012, s.236-247.

[4] Zippel, Richard, Probabilistic algorithms for sparse polynomials. Lecture Notes in Computer Science. 72. pp. 216–226, 1979

[5] J. de Kleine, M. Monagan and A. Wittkopf, Algorithms for the non-monic case of the sparse modular GCD algorithm. Proceedings of ISSAC '05, ACM Press, pp. 124–131, 2005.

[6] Yang, Suling. Computing the Greatest Common Divisor of Multivariate Polynomials over Finite Fields. http://www.cecm.sfu.ca/CAG/theses/suling.pdf

[7] The Berlekamp-Massey Algorithm revisited, N. B. Atti, G. M. Diaz–Toca, H. Lombardi, 9 March 2006

[8] A Unified Approach to HGCD Algorithms for polynomials and integers by Klaus Thull , Chee K. Yap
[9] Factoring polynomials over global fields Belabas, Karim; van Hoeij, Mark; Klüners, Jürgen; Steel, Allan Journal de théorie des nombres de Bordeaux, Volume 21 (2009) no. 1, p. 15-39
[10] P. S. Wang, The EEZ-GCD Algorithm, ACM SIGSAM Bulletin 14, pp. 50–60, 1980
[11] P. S. Wang, An improved multivariate polynomial factoring algorithm. Mathematics of Computation 32, no. 144, 1215–1231, 1978
[12] S. Gao, Absolute irreducibility of polynomials via Newton polytopes, Journal of Algebra 237 (2001), 501–520.
[13] S. Gao and A.G.B. Lauder, Decomposition of polytopes and polynomials, Discrete and Computational Geometry 26 (2001), 89–104.
[14] Jørgen Cherly, Luis Gallardo, Leonid Vaserstein and Ethel Wheland: Solving Quadratic Equations over Polynomial Rings of Characteristic Two. Publicacions Matemàtiques, Vol. 42, No. 1 (1998), pp. 131-142
[15] E. Kaltofen. Sparse Hensel lifting. In EUROCAL 85 European Conf. Comput. Algebra Proc. Vol. 2, pages 4–17, 1985

10. Pseudocode

10.1. **gcd.** For the dense gcd over finite fields, if one runs out of primes of the form $x-\alpha$, instead of failing it is possible to use any irreducible polynomial in place of $x-\alpha$ in Algorithm 1, and this would constitute the large prime version of the algorithm.

Algorithm 1: brownp dense gcd over finite field

```
Input:
    (1) $A, B \in \mathbb{F}_q[x][x_1, \dots, x_n]$ neither is zero
Output:
    (1) monic $G = \gcd(A, B)$, $\bar{A} = A/G$, $\bar{B} = B/G$
if $n = 0$ then return using univariate arithmetic
set $cA = \operatorname{cont}_{x_1,\dots,x_n}(A)$ and $cB = \operatorname{cont}_{x_1,\dots,x_n}(B) \in \mathbb{F}_p[x]$
set $A = A/cA$ and $B = B/cB$                    // content $cA, cB, \dots$ is always monic
set $cG = \gcd(cA, cB)$, $c\bar{A} = cA/cG$ and $c\bar{B} = cB/cG$
set $\gamma = \gcd(\operatorname{lc}_{x_1,\dots,x_n}(A), \operatorname{lc}_{x_1,\dots,x_n}(B)) \in \mathbb{F}_q[x]$
set $bound = 1 + \deg_x \gamma + \max(\deg_x(A), \deg_x(B))$, and set $m = 1 \in \mathbb{F}_p[x]$
pick a prime:                                               // primes are $(x-\alpha)$
choose a new $\alpha \in \mathbb{F}_q$ else return FAIL
set $\gamma^* = \gamma \bmod (x-\alpha)$
set $A^* = A \bmod (x-\alpha)$ and $B^* = B \bmod (x-\alpha) \in \mathbb{F}_q[x_n][x_1, \dots, x_{n-1}]$
if $\gamma^* = 0$ then goto pick a prime
set $(G^*, \bar{A}^*, \bar{B}^*) = $ brownp$(A^*, B^*)$ or goto pick a prime if the call failed
if $G^* = 1$ then set $G = 1, A = \bar{A}, B = \bar{B}$, goto put content
if $\deg_x(m) > 0$ then
    if $\operatorname{lm}_{x_1,\dots,x_n}(G^*) < \operatorname{lm}_{x_1,\dots,x_n}(G)$ then set $m = 1$
    if $\operatorname{lm}_{x_1,\dots,x_n}(G^*) > \operatorname{lm}_{x_1,\dots,x_n}(G)$ then goto pick a prime
end
set $\bar{A} = \operatorname{crt}(\bar{A} \bmod m,\ \bar{A}^* \bmod (x-\alpha))$ and $\bar{B} = \operatorname{crt}(\bar{B} \bmod m,\ \bar{B}^* \bmod (x-\alpha))$
if $\bar{A}$ did not change and, with $T = \bar{A}/\operatorname{cont}_{x_1,\dots,x_n}(\bar{A})$, $T \mid A$ and $A/T \mid B$ then
    set $G = A/T$, $\bar{A} = T$ and $\bar{B} = B/G$, goto fix lcs
end
set $G = \operatorname{crt}(G \bmod m,\ \gamma^* \cdot G^* \bmod (x-\alpha))$ and $m = m \cdot (x-\alpha)$
if $G$ did not change and, with $T = G/\operatorname{cont}_{x_1,\dots,x_n}(G)$, $T \mid A$ and $T \mid B$ then
    set $G = T$, $\bar{A} = \bar{A}/G$ and $\bar{B} = B/G$, goto fix lcs
end
if $\deg_x(m) < bound$ then goto pick a prime
if $\deg_x \gamma + \deg_x A = \deg_x G + \deg_x \bar{A}$ and $\deg_x \gamma + \deg_x B = \deg_x G + \deg_x \bar{B}$ then goto
  success
set $m = 1$, goto pick a prime
success:
set $G = G/\operatorname{cont}_{x_1,\dots,x_n}(G)$, $A = A/\operatorname{lc}_{x_1,\dots,x_n}(G)$ and $B = B/\operatorname{lc}_{x_1,\dots,x_n}(G)$
put content:
set $G = G \cdot cG$, $\bar{A} = \bar{A} \cdot c\bar{A}$ and $\bar{B} = \bar{B} \cdot c\bar{B}$
return $(G, \bar{A}, \bar{B})$
fix lcs:
with $\delta = \operatorname{lc}_{x,x_1,\dots,x_n}(G)$, set $G = \delta^{-1}G$, $A = \delta A$ and $B = \delta B$, goto put content
```

On lines 18 and 22, the inputs $G, \bar{A}, \bar{B}$ are undefined only when $m = 1$, in which case the crt ignores them anyways. There should also be a check analogous to line 19 for the stabilization of $\bar{B}$. This was omitted simply due to space constraints. Finally, the stability checks in lines 19 and 23 (and the missing one for $\bar{B}$) are completely optional and may be executed or skipped on every iteration at the user's discretion.

Similarly to the previous algorithm, divisibility checks could be performed over the integers as well.

Algorithm 2: brownm dense gcd over integers

```
Input: $n \geq 1$
    (1) $A, B \in \mathbb{Z}[x_1, \ldots, x_n]$ neither is zero
Output:
    (1) unit normal $G = \gcd(A, B)$, $\bar{A} = A/G$, $\bar{B} = B/G$
1  set $cA = \mathrm{cont}_{x_1,\ldots,x_n}(A)$ and $cB = \mathrm{cont}_{x_1,\ldots,x_n}(B) \in \mathbb{Z}$
2  set $A = A/cA$ and $B = B/cB$                    // content $cA, cB, \ldots$ is always positive
3  set $cG = \gcd(cA, cB)$, $c\bar{A} = cA/cG$ and $c\bar{B} = cB/cG$
4  set $\gamma = \gcd(\mathrm{lc}_{x_1,\ldots,x_n}(A), \mathrm{lc}_{x_1,\ldots,x_n}(B)) \in \mathbb{Z}[x]$
5  set $bound = 2 \cdot \gamma \cdot \max(|A|_\infty, |B|_\infty)$, and set $m = 1 \in \mathbb{Z}$
6  pick a prime:                                          // primes are numbers
7  choose a new prime $p \in \mathbb{Z}$ else return FAIL
8  set $\gamma^* = \gamma \bmod p$
9  set $A^* = A \bmod p$ and $B^* = B \bmod p \in \mathbb{F}_p[x_n][x_1, \ldots, x_{n-1}]$
10 if $\gamma^* = 0$ then goto pick a prime
11 set $(G^*, \bar{A}^*, \bar{B}^*) =$ brownp$(A^*, B^*)$ or goto pick a prime if the call failed
12 if $G^* = 1$ then set $G = 1, A = \bar{A}, B = \bar{B}$, goto put content
13 if $m > 1$ then
14 |   if $\mathrm{lm}_{x_1,\ldots,x_n}(G^*) < \mathrm{lm}_{x_1,\ldots,x_n}(G)$ then set $m = 1$
15 |   if $\mathrm{lm}_{x_1,\ldots,x_n}(G^*) > \mathrm{lm}_{x_1,\ldots,x_n}(G)$ then goto pick a prime
16 end
17 set $\bar{A} = \mathrm{crt}(\bar{A} \bmod m,\ \bar{A}^* \bmod p)$ and $\bar{B} = \mathrm{crt}(\bar{B} \bmod m,\ \bar{B}^* \bmod p)$
18 set $G = \mathrm{crt}(G \bmod m,\ \gamma^* \cdot G^* \bmod p)$ and $m = m \cdot p$
19 if $m < bound$ then goto pick a prime
20 set $hA = \min(|G|_1 \cdot |\bar{A}|_\infty, |G|_\infty \cdot |\bar{A}|_1)$        // upper bound on $|G \cdot \bar{A}|_\infty$
21 set $hB = \min(|G|_1 \cdot |\bar{B}|_\infty, |G|_\infty \cdot |\bar{B}|_1)$        // upper bound on $|G \cdot \bar{B}|_\infty$
22 if $hA < m$ and $hB < m$ then goto success
23 goto pick a prime
24 success:
25 set $G = G/\mathrm{cont}_{x_1,\ldots,x_n}(G)$, $A = A/\mathrm{lc}_{x_1,\ldots,x_n}(G)$ and $B = B/\mathrm{lc}_{x_1,\ldots,x_n}(G)$
26 put content:
27 set $G = G \cdot cG$, $\bar{A} = \bar{A} \cdot c\bar{A}$ and $\bar{B} = \bar{B} \cdot c\bar{B}$
28 return $(G, \bar{A}, \bar{B})$
```

10.2. **factoring.** The lifting algorithms with be stated with 3 factors.

Algorithm 3: hlift (Multivariate Hensel Lifting - Quintic version)

Input: $m \geq 2$

(1) $(\alpha_1, \dots, \alpha_m) \in R^m$
(2) $A \in R[x_1, \dots, x_m][X]$ with $A(X, \alpha_1, \dots, \alpha_m)$ squarefree
(3) $(B_1, B_2, B_3) \in R[x_1, \dots, x_m][X]$ (however, all but the leading coefficients of each B_i are in $R[x_1, \dots, x_{m-1}]$) such that $A(X, x_1, \dots, x_{m-1}, \alpha_m) = (B_1 B_2 B_3)(X, x_1, \dots, x_{m-1}, \alpha_m)$

Output:

(1) $(B_1, B_2, B_3) \in R[x_1, \dots, x_m][X]$ such that $A(X, x_1, \dots, x_m) = (B_1 B_2 B_3)(X, x_1, \dots, x_m)$ or FAIL

```
1  set $e = A - B_1 B_2 B_3$                                        // current error
2  set $\beta_i = B_i(X, x_1, \dots, x_{m-1}, \alpha_m) \in R[x_1, \dots, x_{m-1}][X]$
3  for $j = 1$ to $\deg_{x_m}(A)$ do
4      assert that $e$ is divisible by $(x_m - \alpha_m)^j$
5      set $t =$ taylor coefficient of $(x_m - \alpha_m)^j$ in $e$      // $t \in R[x_1, \dots, x_{m-1}][X]$
6      $(\delta_1, \delta_2, \delta_3) =$ pfrac$(t, (\beta_1, \beta_2, \beta_3), (\alpha_1, \dots, \alpha_{m-1}), (\deg_{x_1} A, \dots, \deg_{x_{m-1}} A))$
                                   // solve $t = \delta_1\beta_2\beta_3 + \delta_2\beta_1\beta_3 + \delta_3\beta_1\beta_2$
7      if the solved failed then return FAIL
8      set $B_i = B_i + \delta_i (x_m - \alpha_m)^j$ for each $i$
9      set $e = A - B_1 B_2 B_3$
10 end
11 if $e = 0$ then return $(B_1, B_2, B_3)$ else return FAIL
```

Since the solutions δ_i must satisfy $\deg_X \delta_i < \deg_X B_i$, the leading coefficients of the B_i will not be changed by Algorithm 3.

Algorithm 4: hlift (Multivariate Hensel Lifting - Quartic version)

```
Input: $m \geq 2$
    (1) $(\alpha_1, \ldots, \alpha_m) \in R^m$
    (2) $F \in R[x_1, \ldots, x_m][X]$ with $F(X, \alpha_1, \ldots, \alpha_m)$ squarefree
    (3) $(A, B, C) \in R[x_1, \ldots, x_m][X]$ (however, all but the leading coefficients of each $A, B, C$ are
        in $R[x_1, \ldots, x_{m-1}]$) such that $F(X, x_1, \ldots, x_{m-1}, \alpha_m) = (ABC)(X, x_1, \ldots, x_{m-1}, \alpha_m)$
Output:
    (1) $(A, B, C) \in R[x_1, \ldots, x_m][X]$ such that $A(X, x_1, \ldots, x_m) = (ABC)(X, x_1, \ldots, x_m)$ or FAIL
1  set $a_0 = [(x_m - \alpha_m)^0]A$ and set $dA = 0$
2  set $b_0 = [(x_m - \alpha_m)^0]B$ and set $dB = 0$
3  set $c_0 = [(x_m - \alpha_m)^0]C$ and set $dC = 0$
4  for $d = 1$ to $\deg_{x_m}(A)$ do
5      set $t = [(x_m - \alpha_m)^d]F - \sum_{\substack{i+j+k=d \\ i \leq dA,\ j \leq dB,\ k \leq dC}} a_i b_j c_k$
6      use pfrac to find $a_d, b_d, c_d$ from $t = a_d b_0 c_0 + a_0 b_d c_0 + a_0 b_0 c_d$
7      if the solved failed then return FAIL
8      set $a_d = a_d + [(x_m - \alpha_m)^d]A$
9      set $b_d = b_d + [(x_m - \alpha_m)^d]B$
10     set $c_d = c_d + [(x_m - \alpha_m)^d]C$
11     if $a_d \neq 0$ then set $dA = d$
12     if $b_d \neq 0$ then set $dB = d$
13     if $c_d \neq 0$ then set $dC = d$
14     if $dA + dB + dC > \deg_{x_m}(A)$ then return FAIL
15 end
16 assert that $dA + dB + dC = \deg_{x_m}(A)$
17 set $A = \sum_{i=0}^{dA} a_i (x_m - \alpha_m)^i$
18 set $B = \sum_{i=0}^{dB} b_i (x_m - \alpha_m)^i$
19 set $C = \sum_{i=0}^{dC} c_i (x_m - \alpha_m)^i$
20 return (A,B,C)
```

Finally the main work horse. It is easy to solve $t = \delta_1\beta_2\beta_3 + \delta_2\beta_1\beta_3 + \delta_3\beta_1\beta_2$ in $\mathrm{frac}(R)(x_1, \ldots, x_{m-1})[X]$ with pseudo remainder sequences, since $\delta_i = t(\beta_j\beta_k)^{-1} \pmod{\beta_i}$ and check if the δ_i's are defined in $R[x_1, \dots, x_{m-1}][X]$. However, as intermediate expression swell is a problem in this approach. We will use a different algorithm described as below.

Algorithm 5: pfrac (Multivariate partial fraction solver)

```
Input: $l \geq 0$
    (1) $t \in R[x_1, \dots, x_l][X]$
    (2) $(\beta_1, \beta_2, \beta_3)$, where $\beta_i \in R[x_1, \dots, x_l][X]$, $\beta_i$ pairwise coprime in $\mathrm{frac}\,(R)(x_1, \dots, x_l)[X]$
    (3) $(\alpha_1, \dots, \alpha_l) \in R^l$
    (4) $(d_1, \dots, d_l) \in \mathbb{N}^l$ degree bounds
Output:
    (1) $(\delta_1, \delta_2, \delta_3), \delta_i \in R[x_1, \dots, x_r][X]$ such that $t = \delta_1\beta_2\beta_3 + \delta_2\beta_1\beta_3 + \delta_3\beta_1\beta_2$ and
        $\deg_X \delta_i < \deg_X \beta_i$ or *FAIL*
1  if $r = 0$ then
2      set $\delta_i = t(\beta_j\beta_k)^{-1} \pmod{\beta_i}$ in $\mathrm{frac}(R)[X]$
3      if *each* $\delta_i \in R[X]$ then return $(\delta_1, \delta_2, \delta_3)$ else return *FAIL*
4  else
5      set $\tilde{\beta}_i(X) = \beta_i(X, x_1, \dots, x_{r-1}, \alpha_l) \in R[x_1, \dots, x_{l-1}][X]$
6      set $\delta_i = 0$ for each $i$
7      set $e = t$
8      for $j = 0$ to $d_r$ do
9          assert that $e$ is divisible by $(x_r - \alpha_r)^j$
10         set $\tilde{t}$ = taylor coefficient of $(x_r - \alpha_r)^j$ in $e$
11         set $(\tilde{\delta_1}, \tilde{\delta_2}, \tilde{\delta_3}) =$ pfrac$(\tilde{t}, (\alpha_1, \dots, \alpha_{l-1}), (\tilde{\beta}_1, \tilde{\beta}_2, \tilde{\beta}_3), (d_1, \dots, d_{l-1}))$
12         if *the solved failed* then return *FAIL*
13         set $\delta_i = \delta_i + \tilde{\delta}_i(x_r - \alpha_r)^j$
14         set $e = t - (\delta_1\beta_2\beta_3 + \delta_2\beta_1\beta_3 + \delta_3\beta_1\beta_2)$
15     end
16     if $e = 0$ then return $(\delta_1, \delta_2, \delta_3)$ else return *FAIL*
17 end
```