Quantum computation of Gröbner basis through F4 and F5 algorithms

Ichio Kikuchi¹, Akihito Kikuchi^{2*}

¹*Internationales Forschungszentrum für Quantentechnik*

²*International Research Center for Quantum Technology, Tokyo*

March 3, 2022

Abstract

Our main concern in this article is how to apply quantum algorithms to compute Gröbner basis through Faugère’s F4 and F5 algorithms, which are regarded as the most effective algorithms for this purpose. We give examples and pseudo-codes for these algorithms and investigate where we can apply quantum algorithms. As a result, we have the designs (or the pseudo-codes) of quantum versions of the F4 and F5 algorithms by replacing classical algorithms with quantum ones. The quantum versions of F4 and F5 algorithms are, in nature, the extensions of the quantum Gaussian-Jordan elimination of Diepp, since the computations of Gröbner bases are practicable through the matrix representations. In the computation of Gröbner bases, quantum algorithms serve to construct the matrices as small as possible (according to the fundamental idea of F4 and F5) and to find the pivotal element as quickly as possible.

Keywords— Quantum computation; Quantum algorithms; Quantum mechanics; Symbolic computation; Gröbner basis; Buchberger’s algorithm; F4 algorithm, F5 algorithm, F5C algorithm

1 Introduction

In this article, we investigate the quantum computation of Gröbner bases. It is a sequel of our preceding work [KK21], where we have given the general review: we have discussed the applicability of quantum sorting in arranging of polynomials and monomials; we also pointed out the difficulty of quantum linear algorithm which is required to do the reduction of polynomials; We have mentioned how these quantum algorithms could be

*akihito.kikuchi@gakushikai.jp (The corresponding author; a visiting researcher in IFQT)

used in the recent algorithm of Gröbner bases computations, such as F4 [Fau99] and F5 [Fau02a, Fau02a, EP10]. However, the discussion for F4 and F5 was cursory; we had not stepped into details of required quantum computations concerning F4 and F5. Hence, in this article, we minutely inspect the feasibility of quantum computations in each computational step of F4 and F5, assuming that we are the programmers for the computational programs of F4 and F5 on quantum computers. We see how the classical computations are done by classical algorithms and what are the hot spots in the actual computations. Then we investigate the pseudo-codes and find what parts should be replaced with quantum algorithms.

2 Preliminaries

2.1 Quantum Algorithms

We can use the tools of quantum computations as are listed below.

- Quantum search algorithm
- Quantum Maximum (Minimum) findings.
- Quantum Linear Solvers.

The feasibility of these algorithms is investigated in the prequel of this article. So we simply give the following comment.

- A well-known Quantum search algorithm is that of Grover [Gro96], which can find a single element of an unstructured database (with N items) by the time of the order of $\mathcal{O}(\sqrt{N})$.
- The quantum minimum-maximum findings [DH96, AK99] are based on Grover's algorithm. We use this algorithm to find the head terms of polynomials and also for other purposes.
- We have to solve linear problems to do the symbolic computations using linear algebra. We might use the quantum Gauss-Jordan elimination algorithm [DGVM17] or Harrow-Hassidim-Lloyd (HHL) algorithm [HHL09].
- The quantum Gauss-Jordan elimination algorithm uses Grover's search algorithm to find pivotal elements in columns of a matrix. Using the row which holds the pivotal element, the algorithm eliminates the elements in other rows through the classical way of standard Gauss-Jordan elimination. The idea of this algorithm is extendable in the case of Gröbner basis, as we shall see later.
- the HHL algorithm aims to solve the linear equation $Ax = b$ by computing $A^{-1}b$ through quantum dynamics. The inverse of the matrix A^{-1} is not computed directly by the algorithm. However, for the eigenvalues of A , $\{\lambda_i\}$, there is a requirement to compute the inverses $\{1/\lambda_i\}$ clandestinely in the HHL algorithm. It is an obstacle

to the application for Gröbner basis computation, where we will frequently meet not-full-rank matrices.

2.2 Polynomial and Algebra

Here we give a summary of the necessary items for the study.

- Polynomial
- Monomial order
- S-polynomial
- Reduction and normal form

Let f be a polynomial:

$$f = \sum C(\alpha_1, \dots, \alpha_n) x_1^{\alpha_1} \cdots x_n^{\alpha_n}$$

We define the set $M(f)$ of monomials of f as

$$M(f) = \{ C(\alpha_1, \dots, \alpha_n) x_1^{\alpha_1} \cdots x_n^{\alpha_n} \mid C(\alpha_1, \dots, \alpha_n) \neq 0 \}.$$

From this set, we define the set $T(f)$ as

$$T(f) = \{ x_1^{\alpha_1} \cdots x_n^{\alpha_n} \mid C(\alpha_1, \dots, \alpha_n) \neq 0 \},$$

which is composed only of the power products of variables.

When we work with polynomials, it is convenient to give the ordering to the monomials. There are several types of orders:

- Lexicographic ordering.

For $x > y > z$, $(x + y + z + 1)^3$ is expanded and arrayed as

$$x^3 + 3x^2y + 3x^2z + 3x^2 + 3xy^2 + 6xyz + 6xy + 3xz^2 + 6xz + 3x + y^3 + 3y^2z + 3y^2 + 3yz^2 + 6yz + 3y + z^3 + 3z^2 + 3z + 1.$$

- Degree reverse lexicographic ordering.

Similarly, $(x + y + z + 1)^3 \Rightarrow$

$$x^3 + 3x^2y + 3xy^2 + y^3 + 3x^2z + 6xyz + 3y^2z + 3xz^2 + 3yz^2 + z^3 + 3x^2 + 6xy + 3y^2 + 6xz + 6yz + 3z^2 + 3x + 3y + 3z + 1.$$

- Other types.

We define the head term $HT(f)$, the head monomial $HM(f)$, and the head coefficient $HC(f)$ for f as follows.

- $HT(f)$ is the maximum of the entries in $T(F)$ with respect to the assumed monomial orderings.

- $HM(f)$ is the maximum of the entries in $M(F)$ with respect to the assumed monomial orderings.
- $HC(f)$ is the coefficient of $HM(f)$.

The s-polynomial of g_1 and g_2 is defined by

$$spol(g_1, g_2) = HC(g_2)u_1g_1 - HC(g_1)u_2g_2$$

where

$$u_j = \frac{lcm(HT(g_1), HT(g_2))}{HT(g_j)}$$

for $j = 1, 2$.

A critical pair of two polynomials f_1 and f_2 is given by

$$(lcm_{12}, t_1, f_1, t_2, f_2)$$

with monomials lcm_{12} , t_1 , and t_2 such that

$$HT(t_1f_1) = HT(t_2f_2) = lcm(HT(f_1), HT(f_2)) = lcm_{12}.$$

The degree of the critical pair is given by the degree of lcm_{12} .

We use the following terminology

- We say f reduces to g modulo p (or $f \xrightarrow{p} g$), if there exist $tinT(f)$ and a power products of variables s such that $s * HT(p) = t$, and

$$g = f - \frac{r}{HC(p)} \cdot s \cdot p,$$

where r is the coefficient of t .

- We say f reduces to g modulo P (or $f \xrightarrow{P} g$) where P is a set of polynomials, if $f \xrightarrow{p} g$ for some $p \in P$.
- We say f is reducible modulo p or P , if f reduces to some g modulo p or P .
- f is top reducible modlo P if there exsits a polynomial g such that f reduces to g and $HT(g) < HT(f)$.
- If f reduces to g modulo P and g is no longer reducible modulo P , we say g is a normal form of f with respect to $\xrightarrow{P}$

For a set of polynomials $\{f_i = \sum_j M_{ij}t_j\}$ given by the coefficients $\{M_{ij}\}$ and the

power products of variables $\{t_j = x_1^{c_1} x_2^{c_2} \cdots x_n^{c_n}\}$, we represent them by a matrix:

$$M = \begin{matrix} & t_1 & t_2 & t_3 & \cdots \\ \begin{matrix} f_{i_1} \\ f_{i_2} \\ \vdots \\ f_{i_l} \\ f_{i_{l+1}} \\ \vdots \\ f_{i_s} \end{matrix} & \begin{pmatrix} \times & \times & \times & \cdots \\ \times & \times & \times & \cdots \\ \vdots & \vdots & \vdots & \cdots \\ \times & \times & \times & \cdots \\ \times & \times & \times & \cdots \\ \vdots & \vdots & \vdots & \cdots \\ \times & \times & \times & \cdots \end{pmatrix} \end{matrix}.$$

This matrix is transformed into a row echelon form through Gauss-Jordan elimination:

$$\tilde{M} = \begin{matrix} & t_1 & t_2 & \cdots & t_k & t_{k+1} & \cdots & t_m \\ \begin{matrix} f_{i_1} \\ f_{i_2} \\ \vdots \\ f_{i_l} \\ f_{i_{l+1}} \\ \vdots \\ f_{i_s} \end{matrix} & \begin{pmatrix} 1 & 0 & \cdots & 0 & \times & \cdots & \times \\ 0 & 1 & \cdots & 0 & \times & \cdots & \times \\ & & \vdots & & \times & \cdots & \times \\ 0 & 0 & \cdots & 1 & \times & \cdots & \times \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ & & \vdots & & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{pmatrix} \end{matrix}.$$

Let F and $\tilde{F}$ be the set polynomials represented by M and $\tilde{M}$ respectively. Later we use the sets defined by

$$\tilde{F}_+ = \{ g \in \tilde{F} \mid HT(g) \notin HT(F) \}.$$

3 Buchberger algorithm

3.1 Pseudo Code of Buchberger algorithm

Let us review the basic algorithm for Gröbner basis computation: Buchberger algorithm [CLO06, CLO13, EH11]. It is composed of the following parts (see Figure 3.1):

- Selection of pairs of polynomials
- Computation of S-polynomials
- The reduction of S-polynomials with respect to the current (intermediate) Gröbner basis. If the result of the reduction is not zero, it is added to the current Gröbner basis.
- If there is no polynomial to be included in the current Gröbner basis, then the computation terminates.

Figure 3.1: A circuit-like illustration of the Buchberger algorithm. The computation runs from the left to the right in the figure. The data representing the polynomials are transported through the lines and processed by the subroutines represented by small panels. To the panel Spol, two polynomials are transported from two horizontal lines above, namely, from two black dots on the vertical segment directed from the panel. Then the panel Spol computes the s-polynomial and transports the result to the right through the line. Panel Red gets a polynomial f_r from the left and another g from the above (from the resource represented by a black dot), computes the reduction of f with respect to f , and sends the results to the right. The current Gröbner basis at any time of the computation can be taken in the vertical segments of the circuit lines. As the computation proceeds, the number of required data lines will grow, but it shall saturate, because the *reduction* by the panel Red shall return zero-polynomial in the later stage of the computation. If a Red returns zero, the line that holds this panel should be discarded. In the end, the reduction for all possible pairs is zero and the computation terminates.

Algorithm 1 Buchberger algorithm

Input: $F = \{f_1, \dots, f_m\} \in \mathbb{F}[\underline{X}] \setminus \{0\}$ **Output:** the Gröbner basis of $\langle F \rangle$

```
1: function BUCHBERGER( $F$ )
2:    $n \leftarrow m$ 
3:    $g_i \leftarrow f_i$  for  $1 \leq i \leq n$ 
4:    $P \leftarrow \{ (g_i, g_j) \mid 1 \leq i \leq n \}$ 
5:   while  $P \neq \emptyset$  do
6:     Select a pair  $(g_i, g_j) \in P$ 
7:      $P \leftarrow P \setminus \{(g_i, g_j)\}$ 
8:      $s \leftarrow s - \text{polynomial}(g_i, g_j)$ 
9:      $h \leftarrow$  the normal form of  $s$  modulo  $\{g_1, \dots, g_n\}$ 
10:    if  $h \neq 0$  then
11:       $n \leftarrow n + 1$ 
12:       $g_n \leftarrow h$ 
13:       $P \leftarrow P \cup \{ (g_i, g_n) \mid 1 \leq i < n \}$ 
14:    end if
15:  end while
16:  return  $\{g_1, \dots, g_n\}$ 
17: end function
```

In this algorithm, we can apply the quantum algorithms for finding minimum to determine the leading terms of polynomials involved in the computation of S-polynomial. In addition, to compute the normal form of polynomials, we can apply the quantum algorithm to find the possible reduction of polynomials. (The details in the reduction shall be discussed later, for F4 and F5 algorithms.)

4 F4 algorithm

4.1 Demonstration of F4 computation

Let us compute the Gröbner basis using F4 algorithm for an ideal defined by

$\text{f3}=\text{a}+\text{b}+\text{c}, \text{f2}=\text{ab}+\text{ac}+\text{bc}, \text{f1}=\text{abc}-1;$

This ideal has a minimal Gröbner basis generated by

$\text{a}+\text{b}+\text{c}, \text{b}^2+\text{bc}+\text{c}^2, \text{c}^3-1.$

We use the lexicographic order ($a > B > c$).

We abbreviate the terms in the following way: $a^2+b^2+c^2 \rightarrow a2+b2+c2$, by placing the exponent just after the alphabetical symbol. This is the shorthand adopted by computer algebra packages.

We start from this point.

```
F={f3,f2,f1}
G={a+b+c}
```

We prepare all possible combinations of pairs from F. Then we investigate them according to the degrees of corresponding s-polynomials. The computation intermittently inserts new polynomials to G, and we should continue to update the list of the pairs of polynomials.

Degree 2 : Pair(a+b+c, ab+ac+bc)

```
P1={Pair(a+b+c, ab+ac+bc)}
L1={(b,a+b+c),(1,ab+ac+bc)}
F1={ab+b2+bc,ab+ac+bc}
```

We compute the s-polynomial between f2 and f2 (given in the pair P1) and reduce it by G. We use the following notations:

- P1: the pair of polynomials, denoted by Pair(p1,p2)
- L1: the pair of polynomials with monomials involved in the computation of the s-polynomial ($u*p1-v*p2$), denoted by $\{(u,p1),(v,p2)\}$.
- F1: The list of polynomials for which we do the reduction. In the beginning, it contains two components in the s-polynomial: $\{u*p1,v*p2\}$.

We use these lists.

```
Done = {ab}
T(F1)= {ab,b2,bc,ac}
```

The monomial ab (in Done) is eliminated in s-polynomial. In T(F1), we gather the monomials in s-polynomial. We should eliminate as many of them as we can through the reduction by G. To this end, we check whether the other monomials in T(F1) could be eliminated likewise. We have only to see if the monomials are divided by the head terms of the polynomials of the current Gröbner basis. If this is the case, namely, being *top-reducible*, we enlarge F1 by the latter polynomial.

Symbolic Processing

```
b2 is in T(F1)\Done. Done <-Done+{b2}={ab,b2};
                                b2 is not top-reducible by G.
bc is in T(F1)\Done. Done <-Done+{bc}={ab,b2,bc};
                                bc is not top-reducible by G.
```



```

ac is in T(F1)\Done. Done <-Done+{bc}={ab,b2,bc,ac};
ac is top-reducible by f3 in G1;
So F1 <- F1+{c*f3}={ab+ac+bc,ab+b2+bc,ac+bc+c2}
T(F1) ={ab,b2,bc,ac,c2}
c2 is in T(F1)\Done. Done <-Done+{bc}={ab,b2,bc,ac,c2};
c2 is not top-reducible by G.

```

In the above, We denote the relative complement of A in B $A \setminus B$. We denote the union of sets $A \cup B$ by $A + B$. Now $\text{Done} = T(F1)$ and all monomials were checked. The Symbolic processing finishes. $F1$ is the list of the polynomials for which we do the reduction through the elimination in a matrix.

We calculate the row echelon of $F1$. We give it by the polynomials for simplicity, but it should be computed by the row operations in a matrix.

Reduction

```

F1={ab+ac+bc,ab+b2+bc,ac+bc+c2} with HT={ab,ac}
--(Row2-Row1,Row3-Row1)-->
{ab+ac+bc,-ac+b2,-ab+c2}
--(Row1+Row3)-->
{ab+ac+bc,ac-b2,ac+bc+c2}
--(-Row2+Row3)-->
Red(F1)={ab+ac+bc,ac-b2,b2+bc+c2}

```

Namely we have done the elimination as follows.

Start:

	ab	ac	b2	bc	c2
f2	1	1	0	1	0
b*f1	1	0	1	1	0
c*f1	0	1	0	1	1

End:

	ab	ac	b2	bc	c2
f2	1	1	0	1	0
f4	0	1	-1	0	0
f5	0	0	1	1	1

In $\text{Red}(F1)$ (the reduced set of the polynomials), only the last one has the term not included in $\text{HT}(F1)$. We put $\text{RedH}(F1) = \{b2+bc+c2\}$ and $G = \{a+b+c, b2+bc+c2\}$.

Degree 3: Pair(a+b+c, abc-1)

At present, there is no pair of degree 2. We go up to degree 3. We investigate the following pairs.

```
P2={Pair(a+b+c, abc-1)}
L2={(bc,a+b+c),(1,abc-1)}
F2={abc+b2c+bc2,abc-1}
```

```
Done = {abc}
T(F2)={abc,b2c,bc2,1}
```

Symbolic Processing

```
b2c is in T(F2)\Done. Done <-Done+{b2c}={abc,b2c};
b2c is top-reducible by b2+bc+c2 in G.
F2<-F2+{c*(b2+bc+c2)}; T(F2)={abc,b2c,bc2,1,c3}
```

```
bc2 is in T(F2)\Done. Done <-Done+{bc2}={abc,b2c,bc2};
bc2 is not top-reducible by G.
1 is in T(F2)\Done. Done <-Done+{1}={abc,b2c,bc2,1};
1 is not top-reducible by G.
c3 is in T(F2)\Done. Done <-Done+{1}={abc,b2c,bc2,1,c3};
c3 is not top-reducible by G.
```

Now Done==T(F2); the Symbolic processing finishes.

We calculate the row Echelon of F2

Reduction

```
F2={abc+b2c+bc2,abc-1,b2c+bc2+c3} with HT{abc,b2c}
-->(Row2-Row1)-->
{abc+b2c+bc2,b2c+bc2+1,b2c+bc2+c3}
-->(Row3-Row2)-->
Red(F2)={abc+b2c+bc2,b2c+bc2+1,c3-1}
```

In Red(F2), only the last one has the term not included in HT(F2). We put RedH(F2)=c3-1 and G={a+b+c,b2+bc+c2,c3-1}.

Now we have gotten all entries in the minimal Gröbner basis of I. However, the computation shall continue to check the remaining possible pair of degrees 3, 4, and 5, and it gets none of the new polynomials for the Gröbner basis. We omit the demonstration other than one case, in which we see how the simplification works.

Degree 3:Pair(ab+ac+bc, abc-1)

```
P3={Pair(ab+ac+bc, abc-1)}
L3={(c,ab+ac+bc),(1,abc-1)}
F3={abc+ac2+bc2,abc-1}
```

```
Done = {abc}
```

$T(F_3) = \{abc, ac^2, bc^2, 1\}$

Symbolic Processing

ac2 is in $T(F_3) \setminus \text{Done}$. $\text{Done} \leftarrow \text{Done} + \{ac^2\} = \{abc, ac^2\};$
ac2 is top-reducible by $a+b+c$ in G .

We do the Simplification of $c^2(a+b+c)$ from the following consideration.

We have to add $c^2(a+b+c)$ to F_3 , while $c^2(a+b+c)$ has already been reduced in the echelon previously shown as $\text{Red}(F_1)$. It would be better to use the reduced one. However, it is ambiguous which polynomial in $\text{Red}(F_1)$ is the reduced form of this polynomial, because the echelon elimination mixes the rows. In the F_4 algorithm, we chose it from the owner of the same head term, namely, $ac-b^2$, in this case. Hence,

$F \leftarrow F + \{ac^2 - b^2c\}$
 $T(F_3) = \{abc, ac^2, bc^2, 1, b^2c\}$

Symbolic Processing (continued)

bc2 is in $T(F_3) \setminus \text{Done}$. $\text{Done} \leftarrow \text{Done} + \{bc^2\} = \{abc, b^2c, bc^2\};$
 b^2c is not top-reducible by G .
1 is in $T(F_3) \setminus \text{Done}$. $\text{Done} \leftarrow \text{Done} + \{1\} = \{abc, b^2c, bc^2, 1\};$
1 is not top-reducible by G .
 b^2c is in $T(F_3) \setminus \text{Done}$. $\text{Done} \leftarrow \text{Done} + \{1\} = \{abc, b^2c, bc^2, 1, b^2c\};$
 b^2c is top-reducible by b^2+bc+c^2 in G .
We add $\{c*(b^2+bc+c^2)\}$ to F .
In the echelon of F , We find the simplification of this polynomial: b^2c+bc^2+1 .
 $F_3 \leftarrow F_3 + \{b^2c+bc^2+1\}$

Reduction

$F_3 = \{abc+ac^2+bc^2, abc-1, ac^2-b^2c, b^2c+bc^2+1\}$ with $HT(F_3) = \{abc, ac^2, bc^2\}$
 $\rightarrow$
 $\{ac^2+bc^2+1, abc-1, ac^2-b^2c, b^2c+bc^2+1\}$
 $\rightarrow$
 $\{ac^2+bc^2+1, abc-1, -b^2c-bc^2-1, b^2c+bc^2+1\}$
 $\rightarrow$
 $\text{Red}(F_3) = \{ac^2+bc^2+1, abc-1, -b^2c-bc^2-1, 0\}$

In $\text{Red}(F_3)$, all polynomials have the head term included in $HT(F_3)$. We put $\text{RedH}(F_3) = \text{None}$ and G is unchanged.

4.2 Pseudo code of F_4

The F_4 algorithm is composed of the following parts:

- Selection of pairs of polynomials
- Computation of S-polynomials
- The reduction of S-polynomials with respect to the current (intermediate) Gröbner basis. If the result of the reduction is not zero, it is added to the current Gröbner basis.
- If there is no polynomial to be included in the current Gröbner basis, then the computation terminates.

The feature of the algorithm lies in the process of reduction. It improves the inefficiency of the Buchberger algorithm.

In the following, the pseudo-codes are taken from [Fau99].

4.3 Basic version of F4

There are two versions of the F4 algorithm [Fau99]: the basic and the improved version. The basic version is another style of Buchberger algorithm with the matrix representation of polynomials. In this section we investigate it.

In the following pseudo-codes, we use the following notations.

- we denote the monomial ordering by $>$.
- Let p_i ($i = 1, 2$) be the members of a pair of polynomials which we use in the computation of an s-polynomial. We denote the critical pair of (p_1, p_2) by

$$Pair(p_1, p_2) := (lcm(p_1, p_2), u_1, p_1, u_2, p_2),$$

where

$$lcm(p_1, p_2) = lcm(HT(p_1), HT(p_2)) = u_1 \cdot HT(p_1) = u_2 \cdot HT(p_2).$$

- We fetch the members of the critical pair by

$$Left(Pair(p_1, p_2)) = u_1 \cdot p_1,$$

and

$$Right(Pair(p_1, p_2)) = u_2 \cdot p_2.$$

- We denote the set of the power products of the variables by T .

Algorithm 2 F4-Main loop

Input: F , a finite subset of $R[\underline{x}]$

Input: Sel , a function $List(Pairs) \rightarrow List(Pairs)$ such that $Sel(l) \neq 0$ if $l \neq 0$.

Output: a finite subset of $R[\underline{x}]$

```
1: function F4( $F$ )
2:    $G \leftarrow F, \tilde{F}_0^+ \leftarrow F$  and  $d \leftarrow 0$ 
3:    $P \leftarrow \{Pair(f, g) \mid f, g \in G \text{ with } f \neq g\}$ 
4:   while  $P \neq \emptyset$  do
5:      $d \leftarrow d + 1$ 
6:      $P_d \leftarrow Sel(P)$ 
7:      $P \leftarrow P \setminus P_d$ 
8:      $L_d \leftarrow Left(P_d) \cup Right(P_d)$ 
9:      $\tilde{F}_d^+ \leftarrow Reduction(L_d, G)$ 
10:    for  $h \in \tilde{F}_d^+$  do
11:       $P \leftarrow P \cup \{Pair(h, g) \mid g \in G\}$ 
12:       $G \leftarrow G \cup \{h\}$ 
13:    end for
14:  end while
15: end function
```

The main function of F4 prepares the pairs of polynomials (at LINE 3) and processes them. At LINE 6, only the pairs (which should contribute to the generation of the Gröbner basis) are chosen. With the current member of Gröbner basis, the chosen pair is placed in a matrix, in which the reduction is done (at LINE 9). The result of the reduction is added to the current Gröbner basis set (at LINE 12) and the new pairs are created (at LINE 11).

Algorithm 3 F4-Reduction

Input: L , a finite subset of $R[\underline{x}]$

Input: G , a finite subset of $R[\underline{x}]$

Output: a finite subset of $R[\underline{x}]$ (possibly an empty set)

```
1: function REDUCTION( $L, G$ )
2:    $F \leftarrow \text{Symbolic Preprocessing}(L, G)$ 
3:    $\tilde{F} \leftarrow \text{Reduction to row echelon form of } F \text{ with respect to } <$ 
4:    $\tilde{F}^+ \leftarrow \{f \in \tilde{F} \mid HT(f) \notin HT(F)\}$ 
5:   return  $\tilde{F}^+$ 
6: end function
```

In F4-reduction, we do the reduction of a matrix into a row echelon form. To this

end, we can apply the quantum Gauss-Jordan elimination algorithm. At LINE 2, by *Symbolic-Preprocessing*, we prepare the polynomials involved in the reduction at LINE 3. At LINE 4, we can apply the quantum database search to elicit the row with the head terms not belonging to the set $HT(F)$. Recall that $HT(F)$ is the collection of head terms of polynomials in F .

Algorithm 4 F4-Symbolic-Preprocessing

Input: L , a finite subset of $T \times R[x]$.

Input: G , a finite subset of $R[x]$

Output: a finite subset of $R[x]$ (possibly an empty set)

```

1: function SYMBOLIC-PREPROCESSING( $L, G$ )
2:    $F \leftarrow \{t * f \mid (t, f) \in L\}$ 
3:    $Done \leftarrow HT(F)$ 
4:   while  $T(F) \neq F$  do
5:      $m \leftarrow$  an element of  $T(F) \setminus Done$ 
6:      $Done \leftarrow Done \cup \{m\}$ 
7:     if  $m$  is top reducible modulo  $G$  then
8:        $m = m' * HT(f)$  for some  $f \in G$  and some  $m' \in G$ 
9:        $F \leftarrow F \cup \{m' * f\}$ 
10:    end if
11:  end while
12:  return  $F$ 
13: end function

```

The function Symbolic-Preprocessing prepares the polynomials involved in the reduction, by checking the head terms. This routine contains an if-clause in a while-loop. However we could not apply the quantum database search to detect the items that satisfy the condition. This is because the size of the $T(F)$ varies incessantly during the while-loop.

Algorithm 5 Selection

Input: P , a list of critical pairs

Output: a list of critical pairs

```

1: function SEL( $P, d$ )
2:    $d \leftarrow \min\{\deg(lcm(p)) \mid p \in P\}$ 
3:    $P_d \leftarrow \{p \in P \mid \deg(lcm(p)) = d\}$ 
4:   return  $P_d$ 
5: end function

```

The function Selection chooses the pairs of polynomials according to the degree of their s-polynomials. In this function, we can apply quantum algorithms for finding the

minimum.

4.4 An improved version of F4

There is an improved version of F4, which is featured by the reuse of polynomials. After the reduction, we have rows other than the one adopted for a new entry of Gröbner basis.

We keep these rows (or polynomials) and use them in the following reductions.

There are new functions or modifications as follows.

Algorithm 6 Improved F4-Main loop (with criteria)

Input: F , a finite subset of $R[\underline{x}]$

Input: Sel , a function $List(Pairs) \rightarrow List(Pairs)$ such that $Sel(l) \neq 0$ if $l \neq 0$.

Output: a finite subset of $R[\underline{x}]$

```

1: function IMPROVED-F4( $F$ )
2:    $G \leftarrow \emptyset, P \leftarrow \emptyset$  and  $d \leftarrow 0$ 
3:    $P \leftarrow \{Pair(f, g) \mid f, g \in G \text{ with } f \neq g\}$ 
4:   while  $F \neq \emptyset$  do
5:      $f \leftarrow first(F)$  (Choose the first element in  $F$ ).
6:      $F \leftarrow F \setminus \{f\}$ 
7:      $(G, P) \leftarrow Update(G, P, f)$ 
8:     while  $P \neq \emptyset$  do
9:        $d \leftarrow d + 1$ 
10:       $P_d \leftarrow Sel(P)$ 
11:       $P \leftarrow P \setminus P_d$ 
12:       $L_d \leftarrow Left(P_d) \cup Right(P_d)$ 
13:       $(\tilde{F}_d^+, F_d) \leftarrow Reduction(L_d, G, (F_i)_{i=1, \dots, d-1})$ 
14:      for  $h \in \tilde{F}_d^+$  do
15:         $P \leftarrow P \cup \{Pair(h, g) \mid g \in G\}$ 
16:         $(G, P) \leftarrow Update(G, P, h)$ 
17:      end for
18:    end while
19:  end while
20:  return  $G$ 
21: end function

```

The improved main function uses the improved version of Reduction at LINE 13 (given in below), by which we get the row echelon form F_d which represents the polynomials. To update the list of the pairs of the polynomials at LINE 16, it uses the function Update. $Left(P_d)$ and $Right(P_d)$ are taken from the two members of a pair of the polynomials. These polynomials are put into a list L_d and go through the reduction. We have only to add the polynomials represented by $\tilde{F}_d^+$ to the current Gröbner basis (at LINES 14–17).

Algorithm 7 Update

Input: a finite subset G_{old} in $R[x]$

Input: a finite subset of critical pairs P_{old} in $R[x]$

Input h a polynomial. **Output:** an updated list of critical pairs.

```
1: function UPDATE( $G_{old}, P_{old}, h$ )
2:   From  $h, G_{old}$  and  $P_{old}$ , rebuild  $G$  and  $P$ :
3:    $G \leftarrow G_{old} + \{h\}$ 
4:    $P \leftarrow P_{old}$ 
5:   for  $p \in P_{old}$  do
6:      $P \leftarrow P + (h, p)$ 
7:   end for
8:   According to some criterion, remove the unnecessary entries in  $G$  and  $P$ .
9:   return ( $G, P$ )
10: end function
```

The function Update checks the pairs of polynomials. According to some criteria, it removes the unnecessary pairs, which have no contribution.

Algorithm 8 Improved F4-Reduction

Input: L , a finite subset of $R[x]$

Input: G , a finite subset of $R[x]$

Input: $\mathcal{F} = (F_i)_{i=1, \dots, d-1}$, finite subsets of $R[x]$

Output: two finite subsets of $R[x]$

```
1: function IMPROVED-REDUCTION( $L, G, \mathcal{F}$ )
2:    $F \leftarrow \text{Symbolic Preprocessing}(L, G, \mathcal{F})$ 
3:    $\tilde{F} \leftarrow \text{Reduction to Row Echelon Form of } F \text{ with respect to } <$ 
4:    $\tilde{F}^+ \leftarrow \{f \in \tilde{F} \mid HT(f) \notin HT(F)\}$ 
5:   return ( $\tilde{F}^+, F$ )
6: end function
```

In F4-reduction, we do the reduction of a matrix to a row echelon form, for which we can apply the quantum Gauss-Jordan elimination algorithm. At LINE 2, by *Symbolic-Preprocessing*, we prepare the polynomials for the reduction at LINE 3, trying to simplify them. At LINE 4, we can apply the quantum database search to elicit the row with the head terms not belonging to the set $HT(F)$. Recall that $HT(F)$ is the collection of head terms of polynomials in F .

Algorithm 9 Improved F4-Symbolic-Preprocessing

Input: L , a finite subset of $T \times R[\underline{x}]$

Input: G , a finite subset of $R[]$

Input: $\mathcal{F} = (F_i)_{i=1}^{d-1}$, finite subsets of $R[\underline{x}]$

Output: a finite subset of $R[\underline{x}]$

```
1: function IMPROVED_SYMBOLIC-PREPROCESSING( $L, G, \mathcal{F}$ )
2:    $F \leftarrow \{\text{mult}(\text{Simplify}(t, f, \mathcal{F}) \mid (t, f) \in L\}$ 
3:    $Done \leftarrow HT(F)$ 
4:   while  $T(F) \neq Done$  do
5:      $m \leftarrow$  an element of  $T(F) \setminus Done$ 
6:      $Done \leftarrow Done \cup \{m\}$ 
7:     if  $m$  is top reducible modulo  $G$  then
8:        $m = m' * HT(f)$  for some  $f \in G$  and some  $m' \in G$ 
9:        $F \leftarrow F \cup \{\text{mult}(\text{Simplify}(m', f, \mathcal{F}))\}$ 
10:    end if
11:  end while
12:  return  $F$ 
13: end function
```

This improved version of Symbolic-Preprocessing reuses the polynomials (generated by the preceding reductions) for the reduction which should be done in the next step. We have to add the polynomial of the form $t \cdot f$ to the matrix for the reduction. But $t \cdot f$ might be equivalent to another polynomial of the form $s \cdot f'$, where f' has already been found by preceding reductions and $s < t$. In this case, $t \cdot f$ is replaced with the simplified polynomial $s \cdot f'$. The notation $\text{mult}(\text{Simplify}(t, f, \mathcal{F}))$ means the product $s \cdot u$ from the returned result (s, u) of the function $\text{Simplify}(\dots)$, which does the simplification for $t \cdot f$.

Algorithm 10 Improved F4-Simplify

Input: $t \in T$, a term

Input: $f \in R[\underline{x}]$, a polynomial

Input: $\mathcal{F} = (F_i)_{i=1,\dots,d-1}$, finite subsets of $R[\underline{x}]$

Output: a non-evaluated product, namely, an element of $T \times R[\underline{x}]$

```
1: function SIMPLIFY( $t, f, \mathcal{F}$ )
2:   for  $u \in$  list of divisors of  $t$  do
3:     if  $\exists j$  ( $1 \leq j < d$ ) such that  $u \cdot f \in F_j$  then
4:        $\tilde{F}_j \leftarrow$  the row echelon form of  $F_j$  with respect to  $<$ 
5:       There exists a unique  $p \in \tilde{F}_j$  such that  $HT(p) = HT(u \cdot f)$ 
6:       if  $u \neq t$  then
7:         return Simplify( $t/u, p, \mathcal{F}$ )
8:       else
9:         return ( $1, p$ )
10:      end if
11:    end if
12:  end for
13:  return ( $t, f$ )
14: end function
```

This routine simplifies a polynomial f using the accumulated sets of polynomials $\{F_i\}$. There might be a polynomial p in F_k for some k such that $HT(p) = HT(u \cdot f)$. In this case, we should use $(t/u)p$ instead of $t \cdot f$ for the reduction. The function executes the simplification of a polynomial to the full, seeking its equivalences in the set of lists F . At the for-loop (at LINE 2) and at the if-clause (at LINE 3) we can apply the quantum database search.

5 F5

5.1 The policy of F5

When we use the simple Buchberger algorithm, we are obliged to do unnecessary reductions of s-polynomials (into zero modulo the Gröbner basis). To avoid this, Faugère proposed to assign signatures for all polynomials involved in the computation; the signature for a polynomial tells us the information of its first ancestor which lies in the initially given ideal.

The idea is summarized as follows.

- Let $f_1, \dots, f_m$ be the generators of the given ideal. They are the first ingredients of the Gröbner basis computation and are equipped with the signatures $F_1, \dots, F_m$.
- As the polynomials involved in the computation are the productions of successive reductions of s-polynomials, they are the descendants of $f_1, \dots, f_m$. For each polynomial p in the computation, we choose one of f_i , say, f_k , when p is descended from $u \cdot f_k$ (the product with a monomial u) after some algebras. We give the polynomial a labeled representation $r = (uF_k, p)$, where uF_k is the signature.
- We define the multiplication by $a \cdot r = ((a \cdot u) F_k, a \cdot p)$.
- We need a criterion to pick up a unique label (F_k) to give a signature of a polynomial p . As every polynomial is generated by the s-polynomial, the choice of the signature must be done at the computations of s-polynomials. In other words, an s-polynomial inherits the signature from one of the parents who is dominant over another. This dominance is determined by a duplex ordering of monomials and signatures $F_1, \dots, F_m$, as is usually done in the case of the computation of Gröbner basis of modules. Later we define an example of this sort of orderings in Algorithm 11.
- We need one more definition. Let p_i ($i = 1, 2$) be polynomials and let r_i ($i = 1, 2$) be their labeled representation. We take the head term of the labeled polynomial to be that of the polynomial component in it. Let us denote the signature of a labeled polynomial r by $\mathcal{S}(r)$. We define the critical pair of (r_1, r_2) by

$$CritPair(r_1, r_2) = (lcm(r_1, r_2), u_1 r_1, u_2 r_2)$$

where

$$lcm(r_1, r_2) = lcm(HT(r_1), HT(r_2)) = u_1 HT(r_1) = u_2 HT(r_2).$$

In addition, for this critical pair, we require

$$\mathcal{S}(u_1 r_1) \succ \mathcal{S}(u_2 r_2).$$

This critical pair yields an s-polynomial with the signature $u_1 \cdot \mathcal{S}(r_1)$.

The merit of the labeled polynomial is as follows.

First: By checking the signatures in a pair of polynomials, we judge the necessity of the computation of its s-polynomial. Let us assume that we prepare two polynomials $u \cdot (t_1 F_1, q_1)$ and $v \cdot (t_2 F_2, q_2)$ to compute an s-polynomial. We have equivalences with respect to the current Gröbner basis:

$$t_i \cdot f_i \equiv q_i \pmod{G} \text{ for } i = 1, 2.$$

It often happens that the current intermediate Gröbner basis has already contained a labeled polynomial of the form $r_l = (v_l F_1, p_l)$, descended from a polynomial $v_l \cdot f_1$ and that the monomial v_l divides $u \cdot t_1$, viz. $s \cdot v_l = u \cdot t_1$. Then $u \cdot q_1$ would be equivalent to $s \cdot p_l$. Thus it is advantageous for us to postpone the computation for the pair of $(t_1 F_1, q_1)$ and $(t_2 F_2, q_2)$; instead, we should do it through the pair $(v_l F_1, p_l)$ and $(t_2 F_2, q_2)$.

Second: we could discard certain polynomials without computing their reductions. Let us assume that we have computed the Gröbner basis of $(f_2, \dots, f_m)$ as G and we try to compute that of the extended ideal $(f_1) + G$. We have to do the reduction of the polynomial of the form $h \cdot f_1$. However, we can neglect it if h is a monomial divisible by the leading term of an element in G .

To grasp this idea, we use trivial relations $f_i f_j - f_j f_i = 0$. We write trivial relations by

$$u(f_2 f_1 - f_1 f_2) + v(f_2 f_3 - f_3 f_2) + w(f_3 f_1 - f_1 f_3) = 0,$$

namely,

$$(u f_2 + v f_3) f_1 = u f_1 f_2 + v f_1 f_3 - w f_2 f_3 + w f_3 f_2,$$

with arbitrary polynomials u , v , and w .

Let us examine the reduction of a general term of the form $h f_1$.

If $h = u f_2 + v f_3$, $h f_1$ is contained in the ideal generated by $G = \{f_2, f_3, \dots\}$. If G is the Gröbner basis, the computation of the reduction is unnecessary.

Let us shed the light on the problem from a different direction. Let us assume h in the above has a monomial divisible by the leading terms of $\{f_2, f_3\}$. Then we have

$$(h + h_r) f_1 = u f_1 f_2 + v f_1 f_3$$

with an extra polynomial h_r such that

$$h + h_r = u f_2 + v f_3,$$

We always find such a h_r that satisfies the above relations. We have $(h + h_r) \rightarrow 0$ through the reduction with respect to G . Hence

$$h f_1 \equiv -h_r f_1 \pmod{G}.$$

As h_r is represented by a sum of monomials $\{t_i \mid i = 1, \dots, M\}$, namely,

$$h_r = -(t_1 + \dots + t_M),$$

we get

$$hf_1 \equiv t_1f_1 + \dots + t_Mf_1 \pmod{G}.$$

It leads that the reduction of hf_1 (modulo G) is dependent on the other reductions of $\{t_if_1 \mid i = 1, \dots, M\}$. It implies that if the latter reductions are properly done, we do not have to do the reduction of hf_1 . Hence we could omit the reduction of hf_1 from the computation.

5.2 Example of F5 Computation

Let us compute the Gröbner basis of an ideal defined by the following three polynomials:

$$\begin{aligned} I[1] &= xz^2 - y^2t \\ I[2] &= x^2y - z^2t \\ I[3] &= yz^3 - x^2t^2 \end{aligned}$$

We use the degree reverse lexicographic ordering with $x > y > z$.

- We denote the polynomials in the current Gröbner basis by $P[i]$ with the index i .
- We denote the polynomials with signature (the rules) by $r[i] := (\text{Signature}, \text{polynomial})$ with the index i . We refer to them by *rules*.
- We denote the critical pairs by $CP[i, j]$ made of $P[i]$ and $P[j]$.
- We denote the signatures of the polynomials $I[1]$, $I[2]$, and $I[3]$ by $F1$, $F2$, and $F3$. We assume the ordering by $F3 \prec F2 \prec F1$.
- Between two signatures u^*F , v^*F' , we define the ordering $(<, =, >)$ by Algorithm 11.

Algorithm 11 The Comparison of signatures

```
1: function COMPARISONOFSIGNATURES((u,F),(v,F'))
2:   if u < v then
3:     (u,F) < (v,F')
4:     return <
5:   end if
6:   if u > v then
7:     (u,F) > (v,F')
8:     return >
9:   end if
10:  if F < F' then
11:    (u,F) < (v,F')
12:    return <
13:  end if
14:  if F > F' then
15:    (u,F) > (v,F')
16:    return >
17:  end if
18:  if v = v and F = F' then
19:    (u,F) = (v,F')
20:    return =
21:  end if
22: end function
```

- We abbreviate the terms in the following way: $xy^3t - z^4t \rightarrow xy3t - z4t$, by placing the exponent just after the alphabetical symbol. This is the shorthand adopted by computer algebra packages.
- We have done the computation using the implementation of F5 by Edler [Ede08].

Iteration 1

We put $P[1] := I[1]$. Its representation as a labeled polynomial is $r[1] := (F1, P[1])$.

Iteration 2

We put $P[2] := I[2]$ and $r[2] := (F2, P[2])$. We have a critical pair $CP[2,1]$ of degree 5. Its s-polynomial generates a polynomial $xy3t - z4t$. Reducing this polynomial by the current Gröbner basis, we put $P[3] = xy3t - z4t$, and $r[3] = (z2 * F2, P[3])$.

We check the critical pairs: $CP[i,3], i=1, \dots, 3$.

- $\text{CP}[2,3]$ is rejected on account of Line 18 of F5-CritPair because one of the members of the critical pair is top-reducible.)

Let us examine this case.

We have $P[2]=x^2y-z^2t$, $P[3]=xy^3t-z^4t$ and $\text{spol}(P[2],P[3])=y^2t^*P[5]-x^*P[3]$. The labeled representation of $x^*P[3]$ is $x^*r[3]=(xz^2*F2, x^*P[3])$. Its signature is xz^2*F2 , and the term xz^2 is divisible by the reading term of $P[1]=xz^2-y^2t$ which has already been in the Gröbner basis. Hence, according to the criteria of F5, this pair is removed from the computation

- $\text{CP}[3,1]$ is accepted. This is a critical pair of degree 7, which yields a polynomial $P[4]:=z^6t-y^5t^2$ (after the reduction of the corresponding s-polynomial). It is labeled by $r[4]=(z^4*F2, P[4])$. Note that, in this place, we check $\text{CP}[1,3]$ and accept $\text{CP}[3,1]$. This is because of the following principle. For a critical pair $\text{CritPair}(r_1, r_2)$, we have two terms u_1 and u_2 such that $u_1*HT(r_1) = u_2*HT(r_2)$. We always set the ordering of the pair in such a way that $\mathcal{S}(u_1 * r_1) \succ \mathcal{S}(u_2 * r_2)$ for the signature $\mathcal{S}(\dots)$, by swapping the entries if needed. In this case, the critical pair $\text{CP}[1,3]$ is $\{(y^3t, r[1]), (z^2, r[2])\}$. As $z^2 > y^3t$, we replace it with $\text{CP}[3,1] = \{(z^2, r[2]), (y^3t, r[1])\}$
- We check the critical pairs $\text{CP}[i,4]$ ($i=1,\dots,3$) and find that $\text{CP}[2,4]$, $\text{CP}[1,4]$, and $\text{CP}[3,4]$ are rejected. This is because we detect the top-reducible factors in the critical pairs.

For example, let us examine $\text{CP}[2,4]$. We have

$$P[2]=x^2y-z^2t; \quad P[4]=z^6t-y^5t^2; \quad \text{spol}(P[2], P[4])=z^6t^*P[2]-x^2y^*P[4].$$

By the labeled representation, for the components of the s-polynomial, we have

$$z^6t^*r[2]=(z^6t^*F2, z^6t^*P[2]); \quad x^2y^*r[4]=(x^2y^*F2, x^2y^*P[4]).$$

The signature for the latter is x^2y^*F2 , and the term x^2y is divisible by the leading term of $P[1]$ that lies in the previous Gröbner basis. Hence, from the criteria of F5, the pair $\text{CP}[2,4]$ is removed from the computation.

As we have finished the processing of all of the critical pairs, we go out of this iteration. We have

$$P[1]=x^2y-z^2t; \quad P[2]=xz^2-y^2t; \quad P[3]=xy^3t-z^4t; \quad P[4]=z^6t-y^5t^2.$$

Iteration 3

We add $P[5]:=I[3]$ to the current Gröbner basis. It is labeled by $r[5]=(F3, P[5])$.

We check the critical pairs: $\text{CP}[i,5]$ ($i=1,\dots,4$).

- $CP[2,5]$ is useless. In this pair, the contribution from $P[2]$ is re-writable by one of the existing rules.

Let us examine this case. (We use $CP[5,2]$ instead of $CP[2,5]$.)

We have $P[5]=yz^3-x^2t^2$, $P[2]=x^2y-z^2t$, and $\text{spol}(P[5],P[2])=x^2P[5]-z^3P[2]$. The labeled representation of $z^3P[2]$ is $z^3r[2]=(z^3F_2, z^2P[2])$. We find it is rewritable by $z^3r[3]=z^3(z^2F_2, P[3])$. Hence, from the criteria of F5, the pair $CP[5,2]$ is removed from the computation. This is because of the equivalence modulo the current Gröbner basis. F5 algorithm tried to replace $u^*r[i]$ by $v^*r[j]$ for ij if the signature of the former is divisible by that of the latter. In other words, if the condition for the signatures is satisfied, the polynomial $u^*r[i]$ is replaced with the $v^*r[j]$ that is the product of a monomial and a newcomer in the Gröbner basis.

- $CP[5,1]$ is accepted.
- $CP[5,3]$ is useless because, in this pair, the contribution from $P[3]$ is re-writable by one of the existing rules. In this case, it is re-writable by a rule related to $P[4]$.
- $CP[5,4]$ is accepted.

Iteration 3: $d=5$

We have pairs $\{ CP[5,1], CP[5,4] \}$, in which we process the pair of degree 5.

- $CP[5,1]$ is a pair of degree 5, which yields $P[6]:=y^3zt-x^3t^2$. This polynomial is labeled by $r[6]=(x^3F_3, P[6])$. We check the critical pairs $CP[6, i]$ ($i=1, \dots, 5$) and accept three pairs $CP[6,2]$, $CP[6,1]$, and $CP[6,3]$. Let us keep in mind $CP[5,4]$ is not processed yet.

Iteration 3: $d=6$

We have pairs $\{ CP[5,4], CP[6,1], CP[6,2], CP[6,3] \}$, in which we process the pair of degree 6: $CP[6,3]$. This pair yields the polynomial $P[7]:=z^5t-x^4t^2$, which is labeled by $r[7]=(x^4F_3, P[7])$. We check the critical pairs $CP[7, i]$ ($i=1, \dots, 6$) and accept two pairs $CP[7,1]$ and $CP[7,4]$.

Iteration 3: $d=7$

We have pairs $\{ CP[5,4], CP[6,1], CP[6,2], CP[6,3], CP[7,1], CP[7,4] \}$, in which we process the pairs of degree 7: $\{ CP[6,2], CP[6,1], CP[7,1], CP[7,4] \}$.

- $CP[7,4]$ yields a polynomial $P[8]:=y^5t^2-x^4zt^2$, which is labeled by $r[8]=(x^4F_3, P[8])$. We check the pairs $CP[i, 8]$ ($i=1, \dots, 7$) and accept none.
- $CP[7,1]$ yields a polynomial $P[9]:=x^5t^2-y^2z^3t^2$, which is labeled by $r[9]=(x^5F_3, P[9])$.
- We check the pairs $CP[i, 9]$ ($i=1, \dots, 8$) and accept none.

- $CP[6,1]$ and $CP[6,2]$ are useless because, in these pairs, the contributions from either member of the pairs are rewritable by the existing rules.

Iteration3: $d=8$

We have pairs $CP[5,4]$ of degree 8, which yields a polynomial $P[10]:=y^6t^2-xy^2zt^4$. It is labeled by $r[10]=z^3t \cdot F_3, P[10]$.

- We check the critical pairs $CP[i, 19]$ ($i=1, \dots, 10$) and accept none.

Then we go out of this iteration and finish the computation. We get 10 polynomials $P[1], \dots, P[10]$ in the Gröbner basis.

In the above computation, we frequently do the following two checks:

- Are the components in the critical pairs top-reducible?
- Are the components of the critical pair rewritable?

These steps require the quick detection of terms that shall satisfy the conditions.

5.3 Pseudo code of F5

In this section, we give the pseudo-codes of F5 and investigate how to apply quantum algorithms therein. In the following, the pseudo-codes are taken from [Fau02b]. We use a global variable r to store the array of labeled polynomials.

Algorithm 12 F5-Main loop

global r , an array of labeled polynomials

global Rules, an array of simplification rules

Input: $F = (f_1, \dots, f_m)$ a sequence of nonzero homogeneous polynomial in $\mathbb{F}[\underline{X}]$

Output: a Gröbner basis of $\langle F \rangle$

```
1: function F5( $F$ )
2:    $m \leftarrow |F|$ 
3:   Rules  $\leftarrow (())_{j=1}^m$ 
4:    $r \leftarrow ()$ 
5:    $r_m \leftarrow (\mathbf{e}_m, HC(f_m)^{-1} f_m)$ 
6:    $G \leftarrow (\emptyset)_{j=1}^m$ 
7:    $G_m \leftarrow m$ 
8:   for  $i \leftarrow m - 1, \dots, 1$  do
9:      $G_i \leftarrow \text{AlgorithmF5}(f_i, i, G)$ 
10:    if  $(\exists k \in G_i) \text{poly}(r_k) = 1$  then
11:      return  $\{1\}$ 
12:    end if
13:  end for
14:  return  $\{\text{poly}(r_k) \mid k \in G_1\}$ 
15: end function
```

- we store the information of labeled polynomials in a list Rules, since we utilize the labeled polynomials to simplify polynomials generated through the computation. The entries of this list designate the sources and the targets of the simplification of polynomials.
- The signatures of the polynomials $(f_1, \dots, f_m)$ are represented by $\{\mathbf{e}_i\}$.
- The signature of a labeled polynomial r_k is given by $\mathcal{S}(r_k)$.

We use the monic polynomials (Line 5). The main body of the computation is executed by the core function AlgorithmF5. In the following, the term *r-indices* means the list of indexes of labeled polynomials contained in some set, provided that all labeled polynomials are stored in a list r with corresponding indexes. We denote a labeled polynomial by r_l and write its polynomial part by $\text{poly}(k)$. The Gröbner basis in each iteration (indexed by i) in the loop at LINE 8 is stored in G_i , where we use index i to indicate that we process the polynomial f_i . Note that we process the polynomials from f_m to f_1 by descending order.

Algorithm 13 F5-Core routine

Input: $f \in \mathbb{F}[\underline{X}]$; $i \in \mathbb{N}$; G' , a sequence of sets of r-indices

Output: a set G' of r-indices with $i \in G'$

```
1: function ALGORITHMF5( $f, i, G$ )
2:    $r_i \leftarrow (\mathbf{e}_i, HC(f)^{-1}f)$ 
3:    $G' \leftarrow G_{i+1} \cup \{i\}$ 
4:    $P \leftarrow \emptyset$ 
5:   for  $j \in G_{i+1}$  do
6:      $P \leftarrow P \cup \text{CritPair}(i, j, i, G)$ 
7:   end for
8:   while  $P \neq \emptyset$  do
9:      $r_i \leftarrow (\mathbf{e}_i, HC(f)^{-1}f)$ 
10:     $P_d \leftarrow \{p \in P \mid \deg(p) = d\}$ 
11:     $P \leftarrow P \setminus P_d$ 
12:     $S_d \leftarrow \text{Spols}(P_d)$ 
13:     $R_d \leftarrow \text{Reduction}(S_d, G', G)$ 
14:    for  $k \in R_d$  do
15:       $P \leftarrow P \cup \bigcup \{\text{CritPair}(k, l, i, G) \mid l \in G'\}$ 
16:       $G' \leftarrow G' \cup \{k\}$ 
17:    end for
18:  end while
19:  return  $G'$ 
20: end function
```

We prepare the critical pairs (LINE 6). Then we go up through the degrees of critical pairs d (LINE 8), by picking out the pairs with this degree (LINE 10,11). We compute s-polynomials (LINE 12) and do the reduction (LINE 13). With the existing polynomials, the polynomial obtained by the reduction generates new critical pairs (LINE 15) and join in the current Gröbner basis (LINE 16). The list G' contains the indexes of the labeled polynomials which lie in the current Gröbner basis. Note that $G = \{G_{i+1}, \dots, G_m\}$, which is the accumulation of the intermediate Gröbner bases computed by past iterations. We have computed G_{i+1} in the last iteration and now we process f_i . Hence we initialize G' according to LINE 3.

Algorithm 14 F5-CritPair

Input: r_k, r_l are labeled polynomials; i is the current iteration to process the polynomial f_i in F5.

Output: The corresponding critical pair.

```
1: function CRITPAIR( $k, l, i, G$ )
2:    $t \leftarrow \text{lcm}(HT(r_k), HT(r_l))$ 
3:    $u_1 \leftarrow t/HT(r_k)$ 
4:    $u_2 \leftarrow t/HT(r_l)$ 
5:   if  $\mathcal{S}(u_1 r_k) \prec \mathcal{S}(u_2 r_l)$  then
6:     (Swap  $k$  and  $l$ )
7:      $k' \leftarrow k$ 
8:      $k \leftarrow l$ 
9:      $l \leftarrow k'$ 
10:     $u_1 \leftarrow t/HT(r_k)$ 
11:     $u_2 \leftarrow t/HT(r_l)$ 
12:  end if
13:  Let  $(t_1, \mathbf{e}_{k_1}) \leftarrow \mathcal{S}(r_k)$ 
14:  Let  $(t_2, \mathbf{e}_{k_2}) \leftarrow \mathcal{S}(r_l)$ 
15:  if  $u_1 t_1$  is top-reducible by  $G_{k_1+1}$  then
16:    return  $\emptyset$ 
17:  end if
18:  if  $u_2 t_2$  is top-reducible by  $G_{k_2+1}$  then
19:    return  $\emptyset$ 
20:  end if
21:  return  $\{(t, u_1, k, u_2, l)\}$ 
22: end function
```

The function CritPair prepares the critical pairs in the current Gröbner basis. It needs the head terms of polynomials, through a function HT. The function HT shall use the algorithm of quantum maximum findings. Also when the function CritPair judges whether a polynomial is top-reducible by others, it uses quantum algorithms such as the maximum finding and the database search. Note that at LINES 15 and 16, G_{k_1+1} and G_{k_2+1} are the *previous states* of Gröbner basis in the computation, since the implementation of Faugère go down from high to low indices of f_i and G_i .

Algorithm 15 F5-Spols

Input: B , a set of critical pairs**Output:** F , a sequence of r-indices

```
1: function SPOLS( $B$ )
2:    $F \leftarrow ()$ 
3:   for  $(t, u, k, v, l) \in B$  do
4:      $c_1 \leftarrow HC(r_k)$ 
5:      $c_2 \leftarrow HC(r_l)$ 
6:      $s \leftarrow u \cdot poly(r_k) - c_1/c_2 \cdot v \cdot poly(r_l)$ 
7:     if  $s \neq 0$  and (not IsREWRITABLE( $u, k$ )) and (not IsREWRITABLE( $v, l$ )) then
8:        $N \leftarrow |r| + 1$ 
9:        $r_N \leftarrow (u\mathcal{S}(r_k), s)$ 
10:      AddRule( $N$ )
11:       $F_{|F|+1} \leftarrow N$ 
12:     end if
13:   end for
14:   sort  $F$  so that  $i < j \Rightarrow \mathcal{S}(r_{F_i}) \prec \mathcal{S}(r_{F_j})$ 
15:   return  $F$ 
16: end function
```

The routine Spols computes the S-polynomials. It needs the head coefficients of polynomials, namely, the coefficient of the head terms of polynomials, through a function HC (and necessarily HT). The function HT shall use the algorithm of quantum maximum findings. For the if-clauses in the loop over the elements in B (LINE 3–12), the quantum database search could be applied.

Algorithm 16 F5-Reduction

Input: T, G' , sets of r-indices

Input: $G = (G_i)_{i=1}^m$, a sequence of sets of r-indices

Output: D , a set of r-indices

```
1: function REDUCTION( $T, G', G$ )
2:    $D \leftarrow \emptyset$ 
3:   while  $T \neq \emptyset$  do
4:     choose  $k \in T$  such that  $\mathcal{S}(r_k)$  is  $\prec$ -minimal among  $\{\mathcal{S}(r_{k'}) \mid k' \in T\}$ 
5:      $T \leftarrow T \setminus \{k\}$ 
6:      $h \leftarrow$  a normal form of  $poly(r_k)$  with respect to  $\{poly(r_j) \mid j \in G'\}$ 
7:      $r_k \leftarrow (\mathcal{S}(r_k), h)$ 
8:      $K, T' \leftarrow \text{TopReduction}(k, G' \cup D, G)$ 
9:      $D \leftarrow D \cup K$ 
10:     $T \leftarrow T \cup T'$ 
11:  end while
12:  return  $D$ 
13: end function
```

The function Reduction does the reduction of labeled polynomials in the set T with respect to the sequence of sets $\{G_i\}$. The reduction of a polynomial is done by the function TopReduction. In the loop (LINE 3–11), we have to find the minimal element, and we can apply the quantum minimum finding.

Algorithm 17 F5-TopReduction

Input: k , a r-index

Input: G' , a set of r-indices

Input: $G = (G_i)_{i=1}^m$, a sequence of sets of r-indices

Output: D, T , two (possibly empty) sets of r-indices

```
1: function TOPREDUCTION( $k, G', G$ )
2:   if  $\text{poly}(r_k) = 0$  then
3:     (We encounter a reduction to zero, and F5 may not terminate.)
4:     return  $\emptyset, \emptyset$ 
5:   end if
6:    $p \leftarrow \text{poly}(r_k)$ 
7:    $J \leftarrow \text{FindReductor}(k, G', G)$ 
8:   if  $J = \emptyset$  then
9:      $p \leftarrow p / HC(p)$ 
10:     $r_k \leftarrow (\mathcal{S}(r_k), p)$ 
11:    return  $\{k\}, \emptyset$ 
12:  else
13:    Let  $j$  be the single element of  $J$ 
14:     $q \leftarrow \text{poly}(r_j)$ 
15:     $u \leftarrow HT(p) / HT(q)$ 
16:     $p \leftarrow p - HC(p) / HT(q) \cdot u \cdot q$ 
17:    if  $u\mathcal{S}(r_j) \prec \mathcal{S}(r_k)$  then
18:       $r_k \leftarrow (\mathcal{S}(r_k), p)$ 
19:      return  $\emptyset, \{k\}$ 
20:    else
21:       $N \leftarrow |r| + 1$ 
22:       $r_N \leftarrow (u\mathcal{S}(r_j), p)$ 
23:      AddRule( $N$ )
24:      return  $\emptyset, \{k, N\}$ 
25:    end if
26:  end if
27: end function
```

The function Top-Reduction does the reduction of the labeled polynomial r_k with the aid of the function FindReductor. The function Top-Reduction either reduces the existing labeled polynomial furthermore (at LINES 17–18) or produces a labeled polynomial with a new index (at LINES 20–24). For the if-clauses in the loop over the elements in J (LINE 13–25), the quantum database search could be applied.

Note that we have not used Top-Reduction in the example computation of the F5 al-

gorithm, since luckily we have not encountered any polynomial reducible by this function.

Algorithm 18 F5-FindReductor

Input: k , an r-index

Input: G' , a set of r-indices

Input: $G = (G_i)_{i=1}^m$, a sequence of sets of r-indices

Output: R , a set of r-indices (possibly empty)

```

1: function FINDREDUCTOR( $k, G', G$ )
2:    $t \leftarrow HT(r_k)$ 
3:   for  $j \in G'$  do
4:      $t' \leftarrow HT(r_j)$ 
5:     Let  $v_j \mathbf{e}_{k_j} = \mathcal{S}(r_j)$ 
6:     if  $t' \mid t$  then
7:        $u \leftarrow t/t'$ 
8:       if  $u\mathcal{S}(r_j) = S(r_k)$  or IsRewritable( $u, j, k$ ) or  $uv_j$  is top-reducible with respect to
        $G_{k_j+1}$  then
9:         continue
10:      else
11:        return  $\{j\}$ 
12:      end if
13:    end if
14:  end for
15:  return  $\emptyset$ 
16: end function

```

The function FindRedutor tries to find the polynomials in $\{G_i\}$ that shall contribute to the reduction of the labeled polynomial r_k . For the if-clauses in the loop (LINE 3–15), the quantum database search could be applied.

Algorithm 19 F5-AddRule

Input: j , an r-index

```

1: function ADDRULE( $j$ )
2:   Let  $(t, \mathbf{e}_i) = \mathcal{S}(r_j)$ 
3:   Rules[ $i$ ]  $\leftarrow$  Rules[ $i$ ] +  $\{(t, j)\}$ 
4: end function

```

Algorithm 20 F5-IsRewritable

Input: $u \in \mathcal{T}$, a power product of the variables

Input: k , an r-index

Output: true or false

```
1: function ISREWRITABLE( $u, k$ )
2:    $k' \leftarrow \text{Rewrite}(u, k)$ 
3:   return  $k \neq k'$  (true or false)
4: end function
```

Algorithm 21 F5-Rewrite

Input: $u \in \mathcal{T}$, a power product of the variables

Input: k , an r-index

Output: an r-index

```
1: function REWRITE( $u, k$ )
2:   Let  $(v, \mathbf{e}_l) \leftarrow \mathcal{S}(r_k)$ 
3:   for  $j \leftarrow 1, \dots, \#\text{Rules}[l]$  do
4:      $(t, j') \leftarrow \text{Rules}[l][j]$  (the  $j$ th entry of  $\text{Rules}[l]$ )
5:     if  $t \mid uv$  then
6:       return  $j'$ 
7:     end if
8:   end for
9:   return  $k$ 
10: end function
```

The functions (AddRule, IsRewritable, and Rewrite) serve to get *rules* which we should use for simplification of polynomials. When a labeled polynomial r_j has the signature $(t, \mathbf{e}_i)$, the list $\text{Rules}[i]$ receives (t, j) . In the function Rewrite, at the loop (at LINES 3–8) to detect the rule that shall satisfy the if-clause (at LINE 5), the quantum database search could be applied.

6 F5C

There is a variant of F5, called F5C, given by Eder and Perry [EP10]. By this algorithm, let us compute the Gröbner basis of the ideal defined by

$$\begin{aligned} I[1] &= xz^2 - y^2t \\ I[2] &= x^2y - z^2t \\ I[3] &= yz^3 - x^2t^2 \end{aligned}$$

We use the degree reverse lexicographic ordering with $x > y > z$.

- We denote the polynomials in the current Gröbner basis by $P[i]$ with the index i .
- We denote the polynomials with signature (the rules) by $r[i] := (\text{Signature}, \text{polynomial})$ with the index i . We refer to them by *rules*.
- We denote the critical pairs by $CP[i,j]$ made of $P[i]$ and $P[j]$.
- We denote the signatures of the polynomials $I[1]$, $I[2]$, and $I[3]$ by $F1$, $F2$, and $F3$.
- As for the ordering of signatures, we use the algorithm 11.
- We abbreviate the terms in the following way: $xy^3t - z^4t \rightarrow xy^3t - z^4t$, by placing the exponent just after the alphabetical symbol. This is the shorthand adopted by computer algebra packages.

Iteration 1

We put $P[1] := I[1]$. Its representation as a labeled polynomial is $r[1] := (F1, P[1])$.

Iteration 2

We put $P[2] := I[2]$ and $r[2] := (F2, P[2])$. We have a critical pair $CP[2,1]$ of degree 5. Its s-polynomial generates a polynomial $xy^3t - z^4t$. Reducing this polynomial by the current Gröbner basis, we put $P[3] = xy^3t - z^4t$, and $r[3] = (z^2 * F2, P[3])$.

We check the critical pairs: $CP[i,3]$, $i=1, \dots, 3$.

- $CP[2,3]$ is rejected on account of Line 18 of F5-CritPair because the contribution from one of the members of the critical pair is top-reducible. It is the same as in the computation by the F5 algorithm)
- $CP[3,1]$ is accepted. This is a critical pair of degree 7, which yields a polynomial $P[4] := z^6t - y^5t^2$ (after the reduction of the corresponding s-polynomial). It is labeled by $r[4] = (z^4 * F2, P[4])$.
- We check the critical pairs $CP[i,4]$ ($i=1, \dots, 3$) and find that $CP[2,4]$, $CP[1,4]$, and $CP[3,4]$ are rejected. This is because we detect the top-reducible factors in the critical pairs.

We have finished iteration 2.

We have the Gröbner basis:

$P[1]=x^2y-z^2t$, $P[2]=xz^2-y^2t$, $P[3]=xy^3t-z^4t$, $P[4]=z^6t-y^5t^2$.

We apply the inter-reduction and get the minimal reduced Gröbner basis:

$P[1]=x^2y-z^2t$, $P[2]=xz^2-y^2t$, $P[3]=xy^3t-z^4t$, $P[4]=z^6t-y^5t^2$.

In the above, no change has occurred in the basis, but it is not the case in general.

Iteration 3

We put $P[5]:=I[3]=yz^3-x^2t^2$ and check the critical pairs.

- $CP[5,1]$ is accepted.
- $CP[5,2]$ and $CP[5,3]$ are rejected because the contributions from one of the members of the critical pair are top-reducible.
- $CP[5,4]$ is accepted.

*Iteration 3: $d=5$ At $d=5$, we have a critical pair $CP[5,1]$. After reduction, it yields $P[6]:=y^3zt-x^3t^2$. This polynomial has the label $(x^*F5, P[6])$.

We check the critical pairs: $CP[i,6]$, $(i=1,\dots,5)$.

- $CP[6, 1]$, $CP[6, 2]$, and $CP[6, 3]$ are accepted. The positions of the pairs are swapped on account of the ordering.
- $CP[4, 6]$ and $CP[5,6]$ are rejected because the contributions from one of the pairs are top-reducible.

We have $CP[5, 4]$, $CP[6, 1]$, $CP[6, 2]$, and $CP[6, 3]$

*Iteration 3: $d=6$ At $d=6$, we have $P[6,3]$. After reduction, it yields $P[7]=z^5t-x^4t^2$. This polynomial has the label $r[7]=(x^2*F5, P[7])$.

We check the critical pairs: $CP[i,7]$, $(i=1,\dots,6)$.

- $CP[5,7]$, $CP[2,7]$, $CP[3,7]$, and $CP[6,7]$ are rejected, because the contributions from one of the pairs are top-reducible.
- $CP[7,1]$ and $CP[7,4]$ are accepted.

We have $CP[5, 4]$, $CP[6, 1]$, $CP[6, 2]$, $CP[7,1]$, and $CP[7,4]$.

Iteration 4: $d=7$

We have 4 critical pairs of degree 7: $CP[6, 1]$, $CP[6, 2]$, $CP[7,1]$, and $CP[7,4]$.

- $SPOLS[7,4]=y^5t^2-x^4zt^2$. After reduction, it yields $P[8]:=y^5t^2-x^4zt^2$
- $SPOLS[7,1]=-x^5t^2+y^2z^3t^2$. After reduction, it yields $P[9]:=y^5t^2-x^4zt^2$
- $SPOLS[6,1]$ is useless because one of the members of the pair is rewritable.
- $SPOLS[6,2]$ is useless because one of the members of the pair is rewritable.

After reduction SPOLS[7,4] yields $P[8]:=y5t2-x4zt2$. It is labeled by $r[8]:=(x2z*F5, P[8])$

We check the critical pairs: $CP[i,8]$, $i=1,...,7$.

- $CP[1,8]$, $CP[4,8]$, $CP[6,8]$, $CP[5,8]$, and $CP[7,8]$ are rejected because the contributions from one of the pairs are top-reducible.
- $CP[2,8]$, $CP[3,8]$ is rejected because the contributions from one of the pairs are rewritable.

After reduction SPOLS[7,1] yields $P[9]:=y5t2-x4zt2$. It is labelled by $r[9]:=(x3*f5, p[9])$.

We check the critical pairs: $CP[i,9]$, $i=1,...,8$. They are rejected because the contributions from one of the pairs are top-reducible.

*Iteration 4: $d=8$ We have a critical pair of degree 8: $CP[5,4]$. After reduction, it yields $y6t2-x2z3t3$.

We check the critical pairs : $CP[i,10]$, $i=1,...,9$. They are rejected because the contributions from one of the pairs are top-reducible.

Then we go out of this iteration and finish the computation. We get 10 polynomials $P[1],...,P[10]$ in the Gröbner basis.

$P[1]=yz3-x2t2$, $P[2]=xz2-y2t$, $P[3]=x2y-z2t$, $P[4]=xy3t-z4t$,
 $P[5]=z6t-y5t2$, $P[6]=y3zt-x3t2$, $P[7]=z5t-x4t2$, $P[8]=y5t2-x4zt2$,
 $P[9]=x5t2-y2z3t2$, $P[10]=y6t2-xy2zt4$.

We apply the inter-reduction and get the minimal reduced basis:

$P[1]=yz3-x2t2$, $P[2]=xz2-y2t$, $P[3]=x2y-z2t$, $P[4]=xy3t-z4t$,
 $P[5]=z6t-y5t2$, $P[6]=y3zt-x3t2$, $P[7]=z5t-x4t2$, $P[8]=y5t2-x4zt2$,
 $P[9]=y5t2-x4zt2$, $P[10]=x5t2-y2z3t2$

6.1 Pseudo Code of F5C

In this section, we investigate the pseudo-code of the F5C algorithm. The places to which we apply quantum algorithms are almost the same as the F5 algorithm. The main function of the F5C algorithm is given below. It uses a function (Inter_Reduce) to get the reduced Gröbner basis from the not-reduced one. Aside from this, the F5C algorithm uses the same functions as are used in the F5. In the following, the pseudo-codes are taken from [EP10]. Note that in the implementation of F5C, there is a remarkable difference from the F5 as follows.

- To process the set of polynomials $\{f_1, \dots, f_m\}$, the original F5 starts from f_m and goes down to f_1 .
- Meanwhile, the F5C starts from f_1 and goes up to f_m .

This difference causes the alternation of arguments and indexes in the loops between the corresponding functions in these two algorithms.

In the following, we use the following global data.

- r : a list of labeled polynomials.
- Rules: a list of simplification rules.
- $<_T$: the monomial ordering.

Algorithm 22 F5C-Basis/C

global data: r , $Rule$, $<_T$

Input: $F = (f_1, f_2, \dots, f_m)$, a sequence of nonzero homogeneous polynomial in $\mathcal{R}^m$

Input: $<$, an admissible ordering

Output: a Gröbner basis of F with respect to $<$

```
1: function BASIS/C( $F, <$ )
2:    $F$  is sorted by the increasing total degree.
3:    $Rule \leftarrow \text{List}(\text{List}())$ 
4:    $r \leftarrow \text{List}()$ 
5:   Append  $(\mathbf{F}_1, f_1 \cdot HC(f_1)^{-1})$  to  $r$ 
6:    $G_{prev} \leftarrow \{1\}$ 
7:    $B \leftarrow \{f_1\}$ 
8:    $i \leftarrow 2$ 
9:   while  $i \leq m$  do
10:    Append  $(\mathbf{F}_{\#r+1}, f_i \cdot HC(f_i)^{-1})$  to  $r$ 
11:     $G_{curr} \leftarrow \text{Incremental\_Basis}(\#r, B, G_{prev})$ 
12:    if  $\exists \lambda \in G_{curr}$  such that  $poly(\lambda) = 1$  then
13:      return  $\{1\}$ 
14:    end if
15:     $G_{prev} \leftarrow \text{Setup\_Reduced\_Basis}(G_{curr})$ 
16:     $B \leftarrow$  the inter reduction of  $\{poly(\lambda) \mid \lambda \in G_{prev}\}$ 
17:     $i \leftarrow i + 1$ 
18:  end while
19:  return  $B$ 
20: end function
```

This routine takes computational steps quite similar to the original F5. It prepares the labeled polynomials sorted by the increasing order of degrees and gives them to the function `Incremental_Basis`. By $\{\mathbf{F}_i\}$, we indicate the signatures of the polynomials prepared at the start of the function `F5C-Incremental_Basis`.

The function `Incremental_Basis` executes the standard *F5* computation to produce the not-reduced Gröbner basis. Through the computation, it generates the set of $[rules]$, namely, the labeled polynomials, which are used to rewrite the polynomials generated thereafter. However, these rules are repealed by the function `Setup_Reduced_Basis` for the next iteration.

In this function, the quantum algorithms are applied in a similar way to the *F5* algorithm. For example, at LINE 7, the quantum minimum finding is applied. At LINE 11, we find the function `Reduction`. This is the same function used in *F5* (Algorithm 16) and quantum algorithms are applied in the same places therein.

Algorithm 23 F5C-Incremental_Basis

global: $r, <_T$

Input: $i \in \mathbb{N}$

Input: B , the inter reduction of $\{poly(k) \mid k \in G_{prev}\}$

Input: a Gröbner basis of $(f_1, f_2, \dots, f_{i-1})$ with respect to $<_T$

Input: $G_{prev} \subset \mathbb{N}$, indices in r of B

Output: G_{curr} , indices in r of a Gröbner basis of $(f_1, f_2, \dots, f_i)$ with respect to $<_T$

```
1: function INCREMENTAL_BASIS( $i, B, G_{prev}$ )
2:    $curr\_idx \leftarrow \#r$ 
3:    $G_{curr} \leftarrow G_{prev} \cup \{curr\_idx\}$ 
4:   Append List () to Rule
5:    $P = \bigcup_{j \in G_{prev}} \text{Critical\_Pair}(curr\_idx, j, i, G_{prev})$ 
6:   while  $P \neq \emptyset$  do
7:      $d \leftarrow \min\{deg(t) \mid (t, k, u, l, v) \in P\}$ .
8:      $P_d \leftarrow \{(t, k, u, l, v) \in P \mid d = \text{degt}\}$ 
9:      $P \leftarrow P \setminus P_d$ 
10:     $S_d \leftarrow SPols(P_d)$ 
11:     $R \leftarrow \text{Reduction}(S_d, G_{curr}, G_{prev})$ 
12:    for  $k \in R$  do
13:       $P \leftarrow P \cup \left( \bigcup_{j \in G_{curr}} \text{Critical\_Pair}(k, j, i, G_{prev}) \right)$ 
14:       $G_{curr} \leftarrow G_{curr} \cup \{k\}$ 
15:    end for
16:  end while
17:  return  $G_{curr}$ 
18: end function
```

Algorithm 24 F5C-Critical_Pair

Global: $<_T$ **Input:** $k, l \in \mathbb{N}$ such that $1 \leq k \leq l \leq \#r$ **Input:** $i \in \mathbb{N}$ **Input:** $G_{prev} \subset \mathbb{N}$, indices in r of a Gröbner basis of $(f_1, f_2, \dots, f_{i-1})$ with respect to $<_T$ **Output:** $\{(t, u, k, v, l)\}$, a set of indices corresponding a critical pair $\{r_k, r_l\}$, used in the computation of a Gröbner basis of $(f_1, f_2, \dots, f_i)$

```
1: function F5C-CRITICAL_PAIR( $i, k, l, G_{prev}$ )
2:    $t_k \leftarrow HM(Poly(k))$ 
3:    $t_l \leftarrow HM(Poly(l))$ 
4:    $t \leftarrow lcm(t_k, t_l)$ 
5:    $u_1 \leftarrow t/t_k$ 
6:    $u_2 \leftarrow t/t_l$ 
7:    $\tau_1 \mathbf{F}_{\nu_1} \leftarrow Sig(k)$ 
8:    $\tau_2 \mathbf{F}_{\nu_2} \leftarrow Sig(l)$ 
9:   if  $\nu_1 = i$  and  $u_1 \cdot \tau_1$  is top-reducible by  $G_{prev}$  then
10:     return  $\emptyset$ 
11:   end if
12:   if  $\nu_2 = i$  and  $u_2 \cdot \tau_2$  is top-reducible by  $G_{prev}$  then
13:     return  $\emptyset$ 
14:   end if
15:   if  $u_1 \cdot \mathcal{S}(k) \prec u_2 \cdot \mathcal{S}(l)$  then
16:     Swap  $u_1$  and  $u_2$ 
17:     Swap  $k$  and  $l$ 
18:   end if
19:   return  $\{(t, k, u_1, l, u_2)\}$ 
20: end function
```

The F5C algorithm uses the function Critical_Pairs to prepare critical pairs. This function is slightly different from the corresponding one (Algorithm 14) in the F5 algorithm.

Algorithm 25 F5C-Setup_Reduced_Basis

global $r, Rule, <_T$

Input: G_{prev} , a list of indices of polynomials in r that corresponds to the Gröbner basis of $(f_1, f_2, \dots, f_i)$

Output: $G_{curr} \in \mathbb{N}$, indices of polynomials in r that corresponds to the reduced Gröbner basis of $(f_1, f_2, \dots, f_i)$

```
1: function SETUP_REDUCED_BASIS( $G_{prev}$ )
2:   Let  $B$  be the inter-reduction of  $\{poly(k) \mid k \in G_{prev}\}$ 
3:    $G_{curr} \leftarrow \{j\}_{j=1}^{\#B}$ 
4:    $r \leftarrow \text{List}\left(\{(\mathbf{F}_j, B_j)\}_{j=1}^{\#B}\right)$ 
5:   return  $G_{curr}$ 
6: end function
```

The function *Setup_Reduced_Basis* receives the Gröbner basis G_{prev} (not being reduced) and transforms it into the reduced one G_{curr} . The old rules (the labeled polynomials) accumulated in the computation of G_{prev} are not handed down to the next step of the computation; they are discarded. Instead, the new labels $\{F_i\}$ are allotted to the polynomials in G_{curr} .

Algorithm 26 Inter_Reduce

Input: F : a list of polynomials.

Output: R : a list of polynomials. If F is a Gröbner basis, R is a minimal reduced Gröbner basis generated from F .

```
1: function INTER_REDUCE( $F, <$ )
2:    $R = \{\}$ 
3:   for  $i = 1$  to  $\#F$  do
4:      $p \leftarrow F[i]$ 
5:     for  $j = 1$  to  $i - 1$  do
6:        $p \leftarrow$  Reduction of  $p$  by  $F[j]$ 
7:     end for
8:     for  $j = i + 1$  to  $\#F$  do
9:        $p \leftarrow$  Reduction of  $p$  by  $F[j]$ 
10:    end for
11:    if  $p \neq F[i]$  then
12:       $F[i] \leftarrow p$ 
13:       $i \leftarrow 1$ 
14:    end if
15:  end for
16:   $i \leftarrow 1$ 
17:  for  $j = 1$  to  $\#F$  do
18:    if  $F[j] \neq 0$  then
19:       $R[i] = 1/\text{HC}(F[j]) * F[j]$ 
20:       $i \leftarrow i + 1$ 
21:    end if
22:  end for
23:  return  $R$ 
24: end function
```

This function receives a list of polynomials and reduces each polynomial with respect to others. The returned result is the set of monic polynomials. We call this process the *inter_reduction*.

7 Discussion

In this article, we have shown how we implement quantum algorithms as the parts of F4 and F5 algorithms to compute Gröbner basis. We have recognized the following things:

- 1 For detecting elements that shall satisfy required conditions, quantum database search would be useful. This sort of computations would often be done for the

operations for monomials, such as the assessment of divisibility or the retrieval of head terms in polynomials.

- 2 The quantum linear solvers (such as the HHL algorithm to solve $Ax = b$) could not find a suitable place for them in the quantum computation of Gröbner basis.
- 3 The sorting of items is required in many places. However, as we have commented in the prequel of this article, the quantum sorting (based upon pairwise comparisons) shall not achieve remarkable qualitative improvement for efficiency.

Let us give more explanations for item 2 above. First, to get the solution of $Ax = b$, the quantum linear solvers of the HHL algorithm have to compute the inverse of eigenvalues of A . Meanwhile, it often happens that the matrices in the computations of Gröbner bases are not null-rank. Second, the HHL algorithm is applied to the real-valued matrices, while the computation of Gröbner basis would be done in any type of number fields. Hence it is better for us to build the general scheme of computation of Gröbner basis in arbitrary number fields. Along this line, we have done investigations in this article. We see that the quantum computations presented in this article are, in a sense, the extension to the quantum Gauss-Jordan elimination by Diep et al. This method employs the quantum database search to find the pivot elements in the matrices, and execute the eliminations of rows by the classical way. Note that we can apply this algorithm to arbitrary number fields. The quantum computation of Gröbner basis explained in the present article employs the quantum database search likewise; the duties of quantum algorithms are to construct the matrices as small as possible and to find the pivoting elements in the matrices. The second duty is common to quantum Gaussian-Jordan elimination, and the first is the unique part in Gröbner basis computation.

There are several algorithms for number theory to construct the basis sets of mathematical objects or to decompose something into smaller items. Might they be instructive to the problem of Gröbner basis computation? The principal tool of these quantum algorithms is the periodicity finding of a function. The latter is usually done through the quantum Fourier transformation that allows us to evaluate something in parallel. This parallelism is achieved by the evaluations of oracle functions over a finite set of natural integers. By the quantum Fourier transformation, some quantum states shall get larger amplitudes than others, and we shall observe them with higher probability. Then we extract the data of integer numbers registered in these quantum states and seek the answer through them. The quantum Fourier transform is computed over a finite set of data, which would often be arrayed in one dimension. This scheme works well for the decomposition of integers, i.e., for the study of factor rings, modulo N [Sho99]. Similarly, we can describe the Gröbner basis through the factor rings of polynomial ideals. In a pity, in general cases, the size of the factor rings by polynomial ideals would not be finite; therein we expect no finite periodicity, and quantum Fourier transformation over finite items is impracticable. (For example, consider the following two rings: $R[x, y]/(x * y, y)$ and $R[x, y]/(x, y)$. The former contains infinite numbers of polynomials

of the form $f(x) = c_0 + c_1 * x + c_2 * x^2 + \dots c_n * x^n$ with arbitrary degrees, even if we neglect the degrees of freedom in the coefficients.) Exceptionally, if the dimension of a polynomial ideal is zero, we have a finite quotient ring from it; a zero-dimensional ring, generated by an ideal of zero-dimension, represents the set of isolated zeros of a set of polynomial equations. Take note here one cruel fact: the zero-dimensionality of an ideal should be judged through its Gröbner basis... We encounter a chicken-and-egg issue: we would like to compute the Gröbner basis of an ideal with the assumption that the latter is zero-dimensional, meanwhile the correct dimension of an ideal cannot be known prior to the Gröbner basis computation! However the zero-dimensional ideals occur everywhere to describe the wide range of problems of scientific computation, wherein we would be able to guess the existence of isolated roots beforehand. So it is not so groundlessly optimistic to investigate the zero-dimensional ideals, for which we might be able to find a full-fledged quantum algorithm to compute Gröbner basis more easily than in other cases.

References

- [AK99] Ashish Ahuja and Sanjiv Kapoor. A quantum algorithm for finding the maximum. *arXiv preprint quant-ph/9911082*, 1999.
- [CLO06] David A Cox, John Little, and Donal O’shea. *Using algebraic geometry*, volume 185. Springer Science & Business Media, 2006.
- [CLO13] David Cox, John Little, and Donal OShea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer Science & Business Media, 2013.
- [DGVM17] Do Ngoc Diep, Do Hoang Giang, and Nguyen Van Minh. Quantum gauss-jordan elimination and simulation of accounting principles on quantum computers. *International Journal of Theoretical Physics*, 56(6):1948–1960, 2017.
- [DH96] Christoph Durr and Peter Hoyer. A quantum algorithm for finding the minimum. *arXiv preprint quant-ph/9607014*, 1996.
- [Ede08] Christian Eder. Singular prototype implementation of f5 algorithm, 2008. <http://www.math.usm.edu/perry/Research/f5ex.lib>.
- [EH11] Viviana Ene and Jürgen Herzog. *Gröbner Bases in Commutative Algebra*, volume 130. American Mathematical Soc., 2011.
- [EP10] Christian Eder and John Perry. F5c: a variant of faugere’s f5 algorithm with reduced gröbner bases. *Journal of Symbolic Computation*, 45(12):1442–1458, 2010.
- [Fau99] Jean-Charles Faugere. A new efficient algorithm for computing gröbner bases (f4). *Journal of pure and applied algebra*, 139(1-3):61–88, 1999.

- [Fau02a] Faug. A new efficient algorithm for computing gröbner bases without reduction to zero. In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, pages 75–83. ACM, 2002.
- [Fau02b] Jean Charles Faugere. A new efficient algorithm for computing gröbner bases without reduction to zero (f 5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, pages 75–83, 2002.
- [Gro96] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.
- [HHL09] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.
- [KK21] Ichio Kikuchi and Akihito Kikuchi. Quantum computation of groebner basis. *OSF Preprints*, 2021. <https://osf.io/mq4p5>.
- [Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.