Quantum computation of Gröbner basis

Ichio Kikuchi¹, Akihito Kikuchi^{2*}

¹*Internationales Forschungszentrum für Quantentechnik*

²*International Research Center for Quantum Technology, Tokyo*

August 9, 2022

Abstract

In this essay, we examine the feasibility of quantum computation of Gröbner basis which is a fundamental tool of algebraic geometry. The classical method for computing Gröbner basis is based on Buchberger’s algorithm, and our question is how to adopt quantum algorithm there. A Quantum algorithm for finding the maximum is usable for detecting head terms of polynomials, which are required for the computation of S-polynomials. The reduction of S-polynomials with respect to a Gröbner basis could be done by the quantum version of Gauss-Jordan elimination of matrices which represents polynomials. However, the frequent occurrence of zero-reductions of polynomials is an obstacle to the effective application of quantum algorithms. This is because zero-reductions of polynomials occur in non-full-rank matrices, for which quantum linear systems algorithms (through the inversion of matrices) are inadequate, as ever-known quantum linear solvers (such as Harrow-Hassidim-Lloyd) require the clandestine computations of the inverses of eigenvalues. Such algorithms should be used in limited situations with the guarantee that the matrices could be inverted. For example, the transformation from the non-reduced Gröbner basis to the reduced one is of this sort, and the quantum algorithms surely achieve the partial speedup of the computations.

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

1 Introduction

There is a wide brief that we are witnessing the dawning of quantum computers. We are impressed by the pomp of working prototypes of quantum computers that claim *quantum*

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

supremacy. These machines stand before us with the assurance that they shall in no time become user-friendly tools in quantum programming environments. If we would like to utilize quantum computers to the full, we should be well acquainted with quantum algorithms and be able programmers in this new area. Up to now, the study of quantum algorithms has been accumulated through these lines [DW19, Jor17]:

- Query and sort: Deutsch–Jozsa algorithm [DJ92],
Simon’s algorithm [Sim97],
Grover’s algorithm [Gro96] and so on.
- Quantum Fourier transform. [Cop02].
- Integer factorization: Shor’s algorithm. [Sho99].
- Solving linear systems [HHL09, DHM⁺18].
- Algebraic and number theoretic algorithms.
- Hybrid algorithm: a juxtaposition of quantum and classical algorithms.

These algorithms are supposed to be executed by quantum circuits, where quantum superposition should be the cause of quantum supremacy, which, as we naively believe, would achieve the exponential speedup of the computations.

Note that the proposed applications of quantum algorithms are mainly in *well-trodden* regions – in other words, quantum computers are regarded as the replacements of classical ones that process the *numeric* computations. Besides them, we might find another field, less popular, but equally important. It is the *symbolic computation*. In any study of mathematical science, we begin with symbols that represent equations or other ideas. So long as we keep ourselves in the world of symbols, our argument is always exact, and it is a bit of merit for rigorous processes of science. However, symbolic representations are often too complicated for human beings. Hence, with the aid of computers, we could rewrite them into more tamed and simpler forms, from which the encoded information reveals itself. In theory and practice, most of the symbolic computations are done by numerical linear algebra running in the backyards. In several problems, however, the complexities of symbolic computations are enormous; they often scale exponentially or double-exponentially (or worse) with the numbers of the involved symbols [ACM84a, ACM84b, Tar98]. In these circumstances, if we could anticipate the exponential speed-up in symbolic computations by quantum computers, it is progress.

One of the most promising algorithms in computer algebra is that of Buchberger [Buc65b, Buc65a, Buc76, EH11, CLO06, CLO13]. The purpose of this algorithm is to equip a set of polynomials with an adequate basis set (Gröbner basis). We should study the Gröbner basis instead of the initial raw set of polynomials, as these two representations have common roots and the Gröbner basis is so well-arranged that the dependency among variables is more clearly presented. The upper bound of the complexity of Buchberger’s algorithm is estimated as

$$2 \left(\frac{d^2}{2} + d \right)^{2^{n-2}}$$

where n is the number of variables, and d is the maximal total degree of the input polynomials [Dub90]. Although this bound is discouragingly large, we often find that Buchberger's algorithm has terminated more speedily without encountering this bound. Moreover, there are improved implementations of Buchberger's algorithm, which use the scheme of divide-and-conquer, by reducing the size of the linear-algebra operations as small as possible [Fau99, Fau02]. In the following sections, we review Buchberger's algorithm and its successors, find where the hard spots of the computations would be, and investigate if the quantum algorithm could mitigate them.

2 Classical algorithms for Gröbner basis

2.1 Preliminaries

At first, let us review preliminary ideas [CLO06, CLO13, EH11, DP13].

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

- Lexicographic order.
- Degree reverse lexicographic order.
- ...Other types...

A polynomial f has the leading term, denoted by $\text{lt}(f)$, which is maximal with respect to the given monomial order. $\text{lt}(f)$ is the product of the initial coefficient $\text{lc}(f)$ and the initial monomial $\text{in}_<(f)$. (The latter is simply a product of the powers of variables without the coefficient). Then we have a standard representation of f with respect to the set of polynomials $(g_1, \dots, g_m)$ in the following way:

$$f = q_1g_1 + q_2g_2 + \dots + q_mg_m + r,$$

provided that

- no monomials of r is contained in the ideal $(\text{in}_<(g_1), \dots, \text{in}_<(g_m))$
- $\text{in}_<(f) \geq \text{in}_<(q_i g_i)$ for all i .

Between two polynomials we define *S-polynomial* by

$$\text{spoly}(f, g) = \frac{\text{lcm}(\text{in}_<(f), \text{in}_<(g))}{\text{lt}(f)} f - \frac{\text{lcm}(\text{in}_<(f), \text{in}_<(g))}{\text{lt}(g)} g$$

where $\text{lcm}(\cdot, \cdot)$ is the least common multiple. The Gröbner basis is defined as follows:

For an ideal I and a monomial order $<$, if a set of elements $\{g_1, \dots, g_m\}$ in I satisfies the following:

$$\text{in}_<(I) = (\text{in}_<(g_1), \dots, \text{in}_<(g_m)),$$

the set (or the ideal) on the right-hand side is called a Gröbner basis of I with respect to the monomial order $<$.

Remark. Regarding the technical terms, there are several ways of definitions, slightly different among authors and confusing for beginners. Instead of *leading terms or initial monomials*, some authors use the phrases, such as *head terms, head monomials, head coefficients* [Fau99, Fau02]. Their definitions are as follows.

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

In this article, we use both of two ways of definitions, according to the descriptions of the original articles. We give the correspondence between two types of definitions in Table 2.1.

		$f = 2x^2 + xy + y^2$
Head term $HT(f)$	Initial monomial $\text{in}_<(f)$	x^2
Head monomial $HM(f)$	Leading term $\text{lt}(f)$	$2x^2$
Head coefficient $HC(f)$	Leading coefficient $\text{lc}(f)$	2

Table 2.1: The correspondence between technical definitions concerning polynomials. With respect to the lexicographic order $x > y$, the rightmost column shows what these definitions signify for a polynomial $f = 2x^2 + xy + y^2$.

2.2 Buchberger's algorithm

The generation of Gröbner basis of an ideal I is done by Buchberger's algorithm, which goes through the following steps:

1. $G := I$ (Initialization). The polynomials in G are indexed as f_i .
2. For every pair of polynomials in G , compute $S_{ij} = \text{spoly}(f_i, f_j)$.
3. Reduce S_{ij} . That is to say, compute the remainder r of S_{ij} through the standard representation with respect to G . If $r \neq 0$, add r to G .
4. Repeat 1-3 until there are no polynomials to be added.
5. G is the Gröbner basis of I .

In addition, when inserting the reduction of S-polynomials of the pairs of polynomials (f, g) in the provisional Gröbner basis, we should use these two criteria [Buc79]:

- Product Criterion: if the greatest common divisor of the leading monomials of f and g equals 1, (namely $\gcd(\text{in}_<(f), \text{in}_<(g)) = 1$), (f, g) can be omitted.
- Chain Criterion: if the provisional Gröbner basis contains some h , such that $\text{in}_<(h)$ divides $\text{lcm}(\text{in}_<(f), \text{in}_<(g))$, the pairs (f, h) and (h, g) could be omitted.

Let us compute the Gröbner basis of the ideal I composed of three polynomials:

$$\begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} = \begin{pmatrix} x + ye \\ xe + y \\ x^2 + y^2 - 1 \end{pmatrix}.$$

This ideal corresponds to the quantum two-states model given by the secular equation

$$\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = e \begin{pmatrix} x \\ y \end{pmatrix}$$

with the normalization $x^2 + y^2 = 1$, where (x, y) and e represent the wavefunction and the energy spectrum respectively.

If we use Buchberger's algorithm, the computation goes as follows. The reduction of polynomials is denoted by $\rightarrow$. The monomial order is lexicographic with the variables $x > y > e$.

- Initialization of the Gröbner basis: $T = (f_1, f_2, f_3)$.
- $\text{spoly}(f_1, f_2) = ye^2 - y \rightarrow ye^2 - y$ ($=: f_4$).
 $T = (f_1, f_2, f_3, f_4)$.
- $\text{spoly}(f_1, f_3) = xye - y^2 + 1 = y \cdot f_2 - 2y^2 + 1 \rightarrow -2y^2 + 1$ ($=: f_5$).
 $T = (f_1, f_2, f_3, f_4, f_5)$.

- $\text{spoly}(f_1, f_4) = xy + y^2e = x \cdot f_1 \rightarrow 0$.
 $T = (f_1, f_2, f_3, f_4, f_5)$.
- $\text{spoly}(f_1, f_5)$, $\text{spoly}(f_2, f_5)$, and $\text{spoly}(f_3, f_5)$ are unnecessary because of the product criterion.
- $\text{spoly}(f_4, f_5) = -y^2 + (1/2)e^2 = (1/2)f_5 + (1/2)(e^2 - 1) \rightarrow (1/2)(e^2 - 1) (= f_6)$.
 $T = (f_1, f_2, f_3, f_4, f_5, f_6)$.
- $\text{spoly}(f_1, f_6)$, is unnecessary because of the product criterion.
- $\text{spoly}(f_2, f_6) = x + ye = f_1 \rightarrow 0$.
 $T = (f_1, f_2, f_3, f_4, f_5, f_6)$.
- $\text{spoly}(f_3, f_6)$ is unnecessary because of the product criterion.
- $\text{spoly}(f_4, f_6) = 0$.
 $T = (f_1, f_2, f_3, f_4, f_5, f_6)$.
- $\text{spoly}(f_5, f_6)$ is unnecessary because of the product criterion.

We get the Gröbner basis given by

$$\begin{pmatrix} x + ye, \\ xe + y, \\ x^2 + y^2 - 1, \\ ye^2 - y, \\ -2y^2 + 1, \\ \frac{1}{2}e^2 - \frac{1}{2} \end{pmatrix}.$$

This basis includes redundant polynomials. By removing them, we get the simple representation:

$$\begin{pmatrix} x + ye, \\ -2y^2 + 1, \\ e^2 - 1 \end{pmatrix}.$$

Note that this algorithm is quite similar to the Gauss-Jordan elimination of matrix. Indeed, if the ideal I is composed only of linear polynomials, we represent them by a matrix with the rows, each of which holds the coefficients of a linear polynomial. Then Buchberger's algorithm achieves the Gauss-Jordan Elimination among rows. In the multi-variate case, the process is more complicated. For a matrix representation of polynomials, we would employ this way.

- Each horizontal row holds the coefficients of a polynomial.
- The horizontal index of the matrix should observe the monomial order. In other words, each vertical column holds the coefficients of the monomials which have the same power-product of variables.

The computation of S-polynomials involves the multiplication of polynomials and monomials, and it gives birth to the new rows (with the redistribution of entries of the original

rows into different positions). The matrix is enlarged repeatedly (by the horizontal shift of elements and the insertion of the rows), and the elimination is partially done at each extension.

The computations of S-polynomial and the subsequent reduction by the provisional Gröbner basis are united to be the Gauss-Jordan elimination of one matrix to the row echelon form, which shall transform M into $\tilde{M}$ in the following way. (Here we denote the power-products of variables by t_i and the polynomials by f_j .)

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

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

2.3 F4

Among the variety of methods to compute Gröbner basis, the F4 algorithm is featured by the use of echelon, which goes through the following steps [Fau99].

Input F: a set of polynomials.

1. Initialization.

$$G := F, \tilde{F}_0^+ := F, d(\text{an integer}) = 0,$$

$$P := \{(f, g) \mid f, g \in G, \text{ such that } f \neq g\}.$$

3. $d \rightarrow d + 1$.

4. Choose a pair $P_d = (f, g)$ from P , and $P \rightarrow P \setminus P_d$.

5. Find terms t_g, t_f such that $HT(t_g \cdot g) = HT(t_f \cdot f)$. Let $L_d = \{(t_g, g), (t_f, f)\}$

6. $\tilde{F}_d^+ := \text{Reduction}(L_d)$.

7. For each $h \in \tilde{F}_d^+$, $P := P \cup \{Pair(h, g) \mid g \in G\}$, and $G := G \cup \{h\}$.
8. Repeat 3-7 while $P_d \neq \emptyset$.

The first technical point of F4 is to pick up the polynomials from Gröbner basis that shall reduce the S-polynomial. To this end, we list up the monomials in the S-polynomials. If the leading monomial of a polynomial in the Gröbner basis divides them, we should take into consideration this polynomial. (We call such situations by the term *top-reducible*.) Then the matrix is composed of the following polynomials

$$\begin{aligned}
& HC(g) \frac{lcm(HT(f), HT(g))}{HT(f)} f, \\
& HC(f) \frac{lcm(HT(f), HT(g))}{HT(g)} g, \\
& t_{j_1} h_{j_1}, \\
& \dots \\
& t_{j_s} h_{j_s},
\end{aligned}$$

where HT are the leading terms, HC are the leading coefficients, and $t_j h_j$ are made from the polynomials h_j in the Gröbner basis, being multiplied by monomials t_j . Note that we start with a choice of the pair of polynomials (f, g) such that the degree of $lcm(HT(f), HT(g))$ would be as small as possible, and then extend the computation incrementally, by using the pairs (which are larger with respect to this degree.)

The second technical point of F4 is as follows. The matrix F , once prepared, shall go through the following process of reduction, and turn into the reduced form $\tilde{F}$, wherein the newly generated rows represent the new polynomials. One of them originated from the S-polynomial, is chosen to extend the provisional Gröbner basis. (To be precise, if the newborn polynomials have leading terms which are not found among the initially given polynomials, those polynomials should be chosen to extend the provisional Gröbner basis.) Let us denote them by $\tilde{F}_+$. The other rows (denoted by $\tilde{F} \setminus \tilde{F}_+$) are also used for the reduction in the next step, as the auxiliaries for the provisional Gröbner basis. The role of this additional set $\tilde{F} \setminus \tilde{F}_+$ is dictated as follows. Let us suppose the pair of polynomials (p, q) is in consideration to compute an S-polynomial, and their head monomials cancel in $a \cdot p - b \cdot q$, where a and b are monomials. If the head monomial of $b \cdot q$ is equal to $c \cdot r$ (with a monomial c and a polynomial r represented in $\tilde{F} \setminus \tilde{F}_+$) we should use the pair (p, r) in the consideration. In the F4 algorithm presented above, the process *Reduction* executes the reduction of matrices, where some polynomials are replaced with more appropriate ones in full consultation with $\tilde{F} \setminus \tilde{F}_+$.

The inefficiency of F4 is that we have to do the zero-reductions of polynomials which has no contribution to the final result. The zero-reduction occurs too frequently and deteriorates the efficiency of the computation. For this issue, the algorithm F5 is one of the answers. It has a criterion to detect useless polynomials, and its computational process could be represented by matrices. This algorithm could not avoid the zero-reduction in all

cases; however, when the input polynomials are *regular sequences*, it can go without zero-reduction, processing full-rank matrix only. In the terminology of linear algebra, the zero-reduction of polynomials means that we treat a non-full-rank matrix, which gives birth to zero rows after the Gauss-Jordan elimination. As we will see later, the occurrence of non-full-rank matrices is an obstacle to the application of the quantum algorithm. Although the original style of F5 could not collaborate with quantum computers, it will be possible that we anticipate the partial speedup by quantum computation in a variant of F5.

Remark. The matrices would be bloated immensely during the computation of Gröbner basis. The following is one of the benchmark tests, called *cyclic-7*, given by the following ideal,

$$\left(\begin{array}{l} a + b + c + d + e + f + g, \\ ab + ag + bc + cd + de + ef + fg, \\ abc + abg + afg + bcd + cde + def + efg, \\ abcd + abcg + abfg + aefg + bcde + cdef + defg, \\ abcde + abcdg + abcfg + abefg + adefg + bcdef + cdefg, \\ abcdef + abcdeg + abcdfg + abcefg + abdefg + acdefg + bcdefg, \\ abcdefg - 1 \end{array} \right).$$

The size of an intermediate matrix for the computation of the Gröbner basis of I amounts to 1557×3838 , in which only 0.47 percent of the entries are non-zero [Fau99].

2.4 F5

The F5 algorithm is more complicated, although it is representable by matrices [Fau02]. The polynomials are labelled by their *signatures* ($\mathcal{S}$) and their substances (p), namely, $r = (\mathcal{S}, p)$. The input polynomials $f_1, \dots, f_m$ are labelled as

$$(\mathbf{e}_1, f_1), \dots, (\mathbf{e}_m, f_m).$$

The S-polynomial between labeled polynomials (r_1, r_2) generates a new polynomial labeled by

$$(u_{12} r_1, HC(r_2) u_{12} \text{poly}(r_1) - HC(r_1) u_{21} \text{poly}(r_2)),$$

where

$$u_{12} = \frac{\text{lcm}(HT(r_1), HT(r_2))}{HT(r_1)}, \quad u_{21} = \frac{\text{lcm}(HT(r_1), HT(r_2))}{HT(r_2)}$$

and $\text{poly}(r_i)$ (with $i = 1, 2$) is the substances of the labeled polynomials. The choice of the signature is dependent on the ordering in the labeled polynomials; in the above example, we assume $u_2 r_2 \prec u_1 r_1$. Since all labelled representations are generated from $\{(\mathbf{e}_i, f_i), i = 1, \dots, m\}$, they are given by the form $(t\mathbf{e}_i, p)$, which signifies that the polynomial p is descended from the polynomial tf_i . Hence, all polynomials generated through the computation carry the information on their origins, and we can see whereby they are

born. We use the signatures to achieve two following purposes.

- If two labeled polynomials are descended from a common ancestor, one of them can be rewritten into another. This sort of rewriting should observe the ordering of labeled polynomials to keep the irreversible traffic of symbolic processing.
- To check the following criterion: in constructing matrices, it is unnecessary to include a polynomial $(t_i \mathbf{e}_i, p)$ after the multiplication by a monomial t_i , if the leading term of $u_1 t_1$ is top-reducible modulo the provisional Gröbner basis.

Remark. A term (a power product of variables) m is top-reducible modulo G if and only if there exists $g \in G$ and $m' \in T$ such that $m = m' \times LT(g)$.

In the F5 algorithm, if the input system is a regular sequence, there is no reduction to zero for S-polynomials. This is an excellency over F4, as we have only to process full-rank matrices.

Remark. A sequence of polynomials $\{f_1, f_2, \dots, f_m\}$ is a regular sequence when

$$f_i \neq 0 \bmod \langle f_1, \dots, f_{i-1} \rangle \text{ for } i = 1, \dots, m - 1,$$

where $\langle f_1, \dots, f_{i-1} \rangle$ is the ideal generated by the entries between the brackets.

3 Quantum algorithm for Gröbner basis

3.1 Hot spots in the computation of Gröbner basis

Consequently, the hot places in the algorithm would be listed as follows.

- How to find the leading term.
- How to process the Gauss-Jordan elimination in the matrices, or how to process the reduction of a polynomial by the provisional Gröbner basis.

As far as the authors' research reaches, we are currently (in 2021) in the following situation.

- The requisite computation for matrices is Gauss-Jordan elimination, for which quantum algorithm exists.
- In the above, the polynomials are stored as unstructured data. The sorting of monomials and the query of head terms is necessary to present the problem by matrices. For these purposes, quantum algorithms are also available.

In the following sections, we assess the feasibility of quantum algorithms in the computation of Gröbner basis.

3.2 Quantum algorithm for finding the maximum and sorting in monomials

In computing S-polynomials and normal forms, we have to find the leading terms of polynomials. According to Dürr and Hoyer [DH96], or, Ahuja and Kapoor [AK99], there is a quantum algorithm for finding the maximum among N items, which requires $\mathcal{O}(\sqrt{N})$ operations, while the classical method for the same problem takes $\mathcal{O}(N)$ steps. Their algorithm uses Grover's search algorithm. It starts from an initial guess of the maximum, through the iteration finds out greater elements than the current candidate, and in the end, arrives at the maximum. It employs the following steps.

- Let $T[0, \dots, N-1]$ be an unsorted table of N distinct items.
- Oracle:

$$f_i(j) = \begin{cases} 1 & (T[j] > T[i]) \\ 0 & (\text{otherwise}) \end{cases}$$

1. Initial guess: choose a index y randomly from the set $\{0, \dots, N-1\}$.
2. Repeat the following processes in $\mathcal{O}(\sqrt{N})$ steps.
 - 2.1 Initialization of the quantum state

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_i |y\rangle|i\rangle.$$

This operation requires $\mathcal{O}(\log N)$ steps by an n -bit Hadamard transformation, where $n = \lceil \log N \rceil$.

- 2.2. Apply Grover's search algorithm for finding the targeted states x that are the states with

$$f_y(x) = 1.$$

- 2.3. Make a measurement on the first register. The result is the new guess for y .

3. Return y . This is the index of the maximum element in the table.

Note that the oracle for deciding the ordering of two monomials would not be so simple computation as in the case of numbers. As we shall discuss later, it requires the comparison of two sequences, and it should detect the first index where entries of two sequences differ.

As for sorting, any quantum sorting algorithm with binary comparison would take at least $\mathcal{O}(N \log N)$ operations for the input list of N items, as is discussed by Hoyer, Neerbek, and Shi [HNS02]. So quantum computers could not beat classical ones in this case.

Remark. Let us recall here Grover's algorithm. Consider a database with N items which are represented by $|1\rangle, \dots, |N\rangle$ on $\lceil \log(N) \rceil$ qubits. We need an oracle function f . This function returns 1 ($f(x) = 1$) if and only if the data x satisfies the criterion of the search. Conventionally we assume that there is one index ω satisfies $f(\omega) = 1$, and this index is the target of the search. We prepare an unitary operator define by

$$U_\omega|x\rangle = \begin{cases} -|x\rangle & (\text{for } f(x) = 1) \\ |x\rangle & (\text{otherwise}) \end{cases}$$

The algorithm goes through the following steps.

1. Initialization. Prepare the superposition:

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{x=N-1} |x\rangle.$$

2. Repeat Grover iteration $r(N)$ times.
 - 2.1 Apply the unitary operation U_ω to $|s\rangle$.
 - 2.2 Apply the diffusion operator $U_s = 2|s\rangle\langle s| - I$ to $|s\rangle$.
3. Measure the quantum states. If we correctly chose the value of $r(N)$, and if N is sufficiently large, the output be ω . The optimum value of $r(N)$ is $\sim \pi\sqrt{N}/4$.

The quantum query algorithm is useful in constructing echelon: from the Gröbner basis we have to find the elements which shall reduce the polynomial, and it is the query to a database.

3.3 Quantum linear systems algorithm for the reduction of matrices

3.3.1 Quantum Gauss-Jordan elimination

The quantum Gauss-Jordan elimination according to Diep and Giang [DGVM17, DNDHG16] is executed with the following steps:

1. Take the first column $[a_{11}, \dots, a_{N1}]$, where each entry is either zero or non-zero. To the first non-zero value, we use quantum Grover's Search Algorithm, which requires $\mathcal{O}(\sqrt{2}^N)$ operations.
2. Produce the first modified row $r_1 = [1, a_{i2}/a_{i1}, \dots, a_{iN}/a_{i1}]$. Register the resulted row as the first row.
3. Multiply the new first row with $-a_{j1}$, and add the result to the j th row.

4. Change N to $N - 1$. Repeat the steps 1,2,and 3 in the downsized the matrix, from which the topmost row and the leftmost column are removed repeatedly. This step is repeated while $N > 0$.
 5. Recover the initial value of N . Eliminate backward entries $a_{N-1,N}, \dots, a_{1,N}$
 6. Change N to $N - 1$, and repeat the step 5. This step is repeated while $N > 0$.
- * In changing N to $N - 1$, one CNOT gate is needed.

This algorithm is for $N \times N$ matrices, and the size of the total operation is evaluated under this assumption.

In the Quantum Gauss-Jordan Elimination Algorithm one needs at most

$$\frac{N(N-1)(2N+1)}{3} + \left[\sqrt{2} \frac{(\sqrt{2}^N - 1)}{\sqrt{2} - 1} \right] \sim \mathcal{O}(2^{N/2})$$

operations. The first term of this evaluation on the left-hand side is the contribution from the linear algebra computation of the classical way. The second term is from Grover's Search algorithm. This term is very large, but it is inevitable so long as one adopts Grover's search. The assumed search problem is the black-box (query) problem as follows.

Suppose we have a function $f : \{0,1\}^N \rightarrow \{0,1\}$. Here $f(x) = 1$ if and only if the data x points satisfies the search criterion. x is the vector computed from $(a_{11}, a_{21} \dots)$; the elements x_i are 0 or 1 according to whether a_{i1} is zero or non-zero. So there are 2^N possibilities.

In the case of Gröbner basis computation, however, we need slight modification. We must represent the polynomials in the rows of the matrix in the beginning. It requires the sorting and the maximum finding of the monomials in those polynomials stored as unstructured data. It appears that the application of quantum Gauss-Jordan Elimination would be unwelcome because the access to the unstructured data and the quantum query requires operations that increase exponentially. In addition, backward elimination is not always necessary.

3.3.2 Harrow-Hassidim-Lloyd algorithm

Concerning this topic, there is a quantum algorithm by Harrow, Hassidim, and Lloyd (HHL), which solves the linear equation $Ax = b$ in $\text{poly}(\log N, \kappa)$ time, where A is a $N \times N$ matrix with the condition number κ . [HHL09]. It seems that, when we deal with the polynomials with real-number coefficients, we could apply this algorithm to the reduction of matrices, by extracting a square matrix A from the matrix and assuming the remaining columns to be vectors b . In preparing A , we must exclude defective columns

that make the matrix singularⁱ, and it is unsuitable for singular matrices.

There is a way to construct the echelon without the reduction to zero, as proposed by the F5 algorithm. This algorithm could detect the unnecessary pairs of polynomials before their S-polynomials are reduced to zero. In addition, if the inputs polynomials satisfy a certain property (which one calls regular sequences), the F5 algorithm could go without zero reduction.

However, the F5C algorithm is not suitable to adopt the HHL algorithm for solving $Ax = b$. This is on account of the following reasons. The F5 algorithm simply makes the triangular forms of matrices for reducing polynomials through the process, which is called *top-reduction*. On the other hand, when we solve $Ax = b$ for reduction, the upper and the lower triangle part of the matrix A are cleared to zeros. The latter treatment would be harmful to the heart of the computational process of the original style of the F5 since it might conflict with the rewriting rules of polynomials which have been established in the preceding stages of the computation.

Regarding this, the F5C algorithm, a modification of the original F5, presents an alternative plan [EP10]. The F5C algorithm retains the virtue of the F5 (which suppresses the occurrence of zero-reductions), and generates the reduced Gröbner basis. (Recall the definition: a Gröbner basis is called reduced if the leading coefficient of each element of the basis is unity; and no monomial in any element of the basis is included in the ideal generated by the leading terms of the other elements of the basis.) The generation of the reduced Gröbner basis also conflicts with the rewriting rules of polynomials already established. Hence, in the F5C the iterative updates of the signatures and rewriting rules are done through the computation; in the original F5, however, they are kept invariant once they have been established.

Therefore, when we do the reduction by computing $x = A^{-1}b$, we should do the successive updates of the signatures and the rewriting rules through the computation, similarly with the F5C algorithm. When we use the classical algorithms, we can record the rewriting rules extracted from operations, namely, from the subtractions between rows. (Concerning the details, check the processes called TOP REDUCTION in [Fau02, SB06, EP10].) However, in the quantum computations, we could not do that, because they skip such conventional steps. That is to say, quantum algorithms are too quick-witted to do such trivial jobs.

Note that the generation of the reduced Gröbner basis from the non-reduced one is also the elimination in matrices. We can do the elimination by solving the linear equation $Ax = b$, by extracting a square matrix A from the matrix and assuming the remaining columns to be vectors b . In this case, we can find a non-singular matrix A , to which the HHL algorithm is applicable. The reduced Gröbner bases are required in various circumstances, and the quantum algorithms would achieve partial speedup in the computations.

ⁱIf we prepare the integer matrix, we can find the rank of the matrix using a fast algorithm mod p to detect the defective columns [FGLM93].

In summary, the feasibility of the quantum HHL algorithm for the construction of echelons is limited. In the F4 algorithm, it is applicable only in the cases where we could solve the linear equation $Ax = b$ with the assurance that A is non-singular: we could realize this convenient situation by applying the F5 criterion in choosing the rows in the matrix. In the F5 algorithm, the reduction of a matrix in *top reduction* should be executed by Gauss-Jordan elimination because we should record the rewriting rules extracted from this process. The F5C algorithm involves the process of turning the non-reduced Gröbner basis into the reduced one, to which the HHL algorithm is surely applicable.

Remark. The HHL algorithm could potentially solve $Ax = b$ when A is ill-conditioned, by inverting the part of b which corresponds to the well-conditioned part of A . Assume that A and b are given by $A = \sum \tilde{\lambda}_i |u_i\rangle\langle u_i|$ and $|b\rangle = \sum_{j=1}^N \beta_j |u_j\rangle$. The essence of the HHL algorithm is the preparation of the following quantum state:

$$\sum_{j=1}^N \beta_j \left(\sqrt{1 - \frac{C^2}{\tilde{\lambda}_j^2}} |\lambda_j\rangle |u_j\rangle |0\rangle + \frac{C}{\tilde{\lambda}_j} |\lambda_j\rangle |u_j\rangle |1\rangle \right).$$

We *uncompute* it into

$$|0\rangle \otimes \sum_{j=1}^N \beta_j \left(\sqrt{1 - \frac{C^2}{\tilde{\lambda}_j^2}} |u_j\rangle |0\rangle + \frac{C}{\tilde{\lambda}_j} |u_j\rangle |1\rangle \right).$$

This representation includes $A^{-1}|b\rangle = \sum_{j=1}^N \frac{\beta_j}{\tilde{\lambda}_j} |u_j\rangle$, which we can observe. The division between *well-conditioned* and *ill-conditioned* part could potentially be done at this point:

$$\sum_j \frac{\beta_j}{\tilde{\lambda}_j} |u_j\rangle |\text{well}\rangle + \sum_l \beta_l |u_l\rangle |\text{ill}\rangle.$$

However, this expediency is merely a theoretical suggestion, and the design of a quantum gate to do this projection is problematic.

Nevertheless, it would be unjust to deny the feasibility of the HHL algorithm. We should keep in mind the following observations.

We can assume that the ill-conditioned part of the matrix contains the eigenvectors with the zero-eigenvalue. Then the expression of the solution of the HHL algorithm (including the well- and ill-conditioned parts as above) essentially gives the result of

Gauss-Jordan Elimination:

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} & b_m \end{array} \right] \Rightarrow \left[\begin{array}{cccccc|c} 1 & 0 & \dots & 0 & 0 & \dots & 0 & c_1 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & c_2 \\ \vdots & \vdots & \ddots & \vdots & 0 & \dots & 0 & \vdots \\ 0 & 0 & \dots & 1 & 0 & \dots & 0 & c_l \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & c_{l+1} \\ \vdots & \vdots & \vdots & \vdots & 0 & \vdots & 0 & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & c_m \end{array} \right]$$

In other words, we can make the echelon form to get the vector $[c_1, c_2, \dots, c_m]$ from the matrix $(a_{i,j})$ and the vector $[b_1, \dots, b_m]$.

To get the well-conditioned part, we could do the projection:

$$\begin{aligned} |b\rangle &= \sum_j b_j \beta_j |u_j\rangle \\ &\Rightarrow \sum_j b_j \beta_j |u_j\rangle |\lambda_j\rangle \\ &\Rightarrow \sum_j b_j \lambda_j \beta_j |u_j\rangle |\lambda_j\rangle = \sum_{j, \lambda_j \neq 0} b_j \lambda_j \beta_j |u_j\rangle |\lambda_j\rangle \\ &\Rightarrow \sum_{j, \lambda_j \neq 0} b_j \lambda_j \frac{1}{\lambda_j} \beta_j |u_j\rangle |\lambda_j\rangle \end{aligned}$$

Once the vector $|b\rangle$ is projected to the range of $A|x\rangle$ (for arbitrary $|x\rangle$), we could apply the HHL algorithm. Likewise, we could get the ill-conditioned part of $|b\rangle$.

Here we see inconveniences which we should get over.

First, the well- and ill-conditioned parts are decomposed in different quantum states, while what we want to get is their sum. In quantum computing, we have to get together these two states from different resources, and for this purpose, we have to contrive additional algorithms and quantum circuits.

Second, the matrix A we have to process in our problem is not hermitian, while the HHL algorithm is applied to hermitian matrices. For non-hermitian cases, we have to solve a modified problem:

$$\begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} 0 \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

The ill-conditioned subspace of the modified hermitian matrix on the left-hand side would be larger than that of the original A . We have to cull the latter through additional quantum computations.

3.4 Quantum algorithm for the related topics

3.4.1 FGLM algorithm

Concerning Buchberger's algorithm, there is an important idea, called the FGLM algorithm [FGLM93]. The efficiency of Buchberger's algorithm strongly depends on the choice of the monomial orders. For example, the computation with degree reverse lexicographic order excels that with normal lexicographic order. So it is probable that higher efficiency would be achieved by computing Gröbner basis with degree reverse lexicographic order, and then by changing the order into lexicographic one, which gives rise to another basis set. The FGLM algorithm does the change of monomial orders using linear algebra, with the following steps. Here we consider the conversion of a Gröbner basis from an arbitrary monomial order to the lexicographic order. We distinguish the old and new basis by G and G_{lex} ; we denote monomials by the abbreviation $X^a = x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}$; we represent its normal form with respect to G by $\overline{X^a}^G$.

1. Initialization: $G_{lex} = B_{lex} = \emptyset$. Choose the initial input monomial as $X^a = 1$.
2. Main Loop: Compute the normal form $\overline{X^a}^G$ of the chosen monomial X^a . We meet two cases:

Case 1. Between X^a and the elements $\{Y^b\} \in B_{lex}$, there is a linear dependence:

$$\overline{X^a}^G + \sum_j c_j \overline{Y^b}^G = \overline{0}^G,$$

where c_j are coefficients in the number field of the base ring. Then add

$$g := X^a + \sum_j c_j Y^b$$

to G_{lex} .

Case 2. X^a and the elements $\{Y^b\} \in B_{lex}$ are independent. Then add X^a to B_{lex} .

3. Termination Test: Let x_m be the greatest variable in the new monomial order. If the head term of g ($HT(g)$) is a power of x_m , the algorithm terminates.
4. Choice of new input: Choose the next input monomial which is not divisible by any of the head terms of the polynomials in G_{lex} .

The key point in the algorithm is the computation of normal forms, namely, the Gauss-Jordan elimination in the matrix, to which the quantum algorithm (introduced in the previous section) is also applicable.

3.4.2 Krull dimension

Recall the definition of Krull dimension.

Let R be a ring. A sequence of prime ideals $p_0 \subsetneq p_1 \subsetneq \dots \subsetneq p_m$ is called a chain of prime ideals of R of length m . The Krull dimension of R , denoted by $\dim(R)$, is the supremum of the lengths of such chains. If $I \subset R$ is an ideal, the Krull dimension of I ($\dim(I)$) is defined to be the dimension of R/I .

In a nutshell, $\dim(I)$ is the dimension of the algebraic set depicted by the common zeros of the polynomials in the ideal I . The most important case for practical purposes is that of zero dimension because the corresponding algebraic set is composed of the discrete zeros of the set of polynomial equations, which are required by various computations of physical science. The Krull dimension of I is computed from its Gröbner basis; to be more precise, it is computed from the ideal generated by its leading monomials [DP13]. Whether the Gröbner basis is computed by quantum or classical way, the leading monomials are found by quantum computation, as has been discussed already.

3.4.3 The monomial orders

In the multivariate case, we encounter the monomials of the form $\dots x^i y^j z^k \dots$, and for the convenience of study, we must give the appropriate order to the monomials.

Let $\mathbf{x}^{\mathbf{a}} (= x_1^{a_1} \dots x_n^{a_n})$ and $\mathbf{x}^{\mathbf{b}} (= x_1^{b_1} \dots x_m^{b_m})$ be the monomials (without coefficients). And let $\mathbf{a}$ and $\mathbf{b}$ be sequence of superscripts $(a_1, a_2, \dots, a_n)$ and $(b_1, b_2, \dots, b_n)$.

There are several options for the orders.

- The lexicographic order.

$$\mathbf{x}^{\mathbf{a}} < \mathbf{x}^{\mathbf{b}} \Leftrightarrow \exists i \ (1 \leq i \leq n) : a_1 = b_1, \dots, a_{i-1} = b_{i-1}, a_i < b_i.$$

This definition is equivalent to the following statement.

$$\mathbf{x}^{\mathbf{a}} < \mathbf{x}^{\mathbf{b}}, \text{ if the leftmost component of } \mathbf{a} - \mathbf{b} \text{ is negative.}$$

- The reverse lexicographic order.

$$\mathbf{x}^{\mathbf{a}} < \mathbf{x}^{\mathbf{b}} \Leftrightarrow \exists i \ (1 \leq i \leq n) : a_n = b_n, \dots, a_{i+1} = b_{i+1}, a_i > b_i.$$

This definition is equivalent to the following statement.

$$\mathbf{x}^{\mathbf{a}} < \mathbf{x}^{\mathbf{b}}, \text{ if the rightmost component of } \mathbf{a} - \mathbf{b} \text{ is positive.}$$

- The degree-reverse-lexicographic order:

$$\text{Let } \deg(\mathbf{x}^{\mathbf{a}}) = a_1 + \dots + a_n.$$

$$\mathbf{x}^{\mathbf{a}} < \mathbf{x}^{\mathbf{b}} \Leftrightarrow$$

$$(1) \deg(\mathbf{x}^{\mathbf{a}}) < \deg(\mathbf{x}^{\mathbf{b}})$$

or

$$(2) \deg(\mathbf{x}^{\mathbf{a}}) = \deg(\mathbf{x}^{\mathbf{b}}) \text{ and } \exists i \ (1 \leq i \leq n) : a_n = b_n, \dots, a_{i+1} = b_{i+1}, a_i > b_i.$$

(The rightmost component of $\mathbf{a} - \mathbf{b}$ is positive.)

The evaluation of inequalities between monomials is not so simple as in the case of numbers. It requires the search over the sequence of data, and if the data is large, the quantum algorithm will work well. We could assume that the oracle shall detect the

ordering between two items with ease, but when we treat lengthy polynomials with many variables, the computational cost of the comparison of monomials would not be negligible.

3.4.4 Inserting an element into an ordered set

During the Gröbner basis computation, we have to insert a newborn polynomial into the list of the provisional Gröbner basis. Concerning this, there is a study by Farhi et al. [FGGS98], which considers the problem of inserting a new item into an ordered list of $N - 1$ items. They have shown this: classical computations determining the insertion point requires $\log(N)$ comparisons, while for N large, no quantum algorithm can reduce the number of comparisons below $\log(N)/(2 \log(\log(N)))$.

3.4.5 Quantum RAM

We have to store the real-valued vectors for the computations. In quantum computers, they should be represented as

$$|x\rangle := \frac{1}{\|\mathbf{x}\|} \sum_{i=1}^N x_i |i\rangle,$$

on $\lceil \log(N) \rceil$ qbits. A polynomial could be represented in this way, in which the coefficients and the monomials are placed at the amplitudes x_i and the quantum states $|i\rangle$. The quantum RAM is the operation that does the following transformation:

$$\sum_{i=1}^N |x_i\rangle |i\rangle \rightarrow \frac{1}{\|\mathbf{x}\|} \sum_{i=1}^N x_i |i\rangle.$$

In other words, it decodes the state $|x_i\rangle$ into the amplitude x_i . The effective method using unitary transformation is proposed by Prakash [Pra14].

3.4.6 Termination

We should be careful in the quantum computation of Gröbner basis. In general, quantum computations are possessed by errors in the observations. We, however, optimistically believe that the observation returns the correct answer when it reaches us with a high probability. On the other hand, the algorithms for computing Gröbner basis assumes the absolute correctness of the involved sub-procedures: they get correct inputs, execute the rigid symbolic computations, and pass the answers to the succeeding steps. One of the most important requirements for symbolic computations is that they should terminate after a finite number of steps, and the termination is usually guaranteed that they shall methodically process all of the intermediate data (given in a well-ordered style), leaving no stones unturned. To use a figurative expression, those algorithms are of a kind of precision machines, and they are not so robust in front of the occurrence of errors. In other words, a trivial local error would cause the disorder of the whole, which thereafter plunges into a never-ending computation. In the present stage, we do not comprehend

sufficiently the character of symbolic computations with inherent probabilistic errors, for which we have to develop efficient tactics of quantum error detection and correction.

4 Summary and Discussion

The feasibility of quantum algorithms to compute Gröbner basis is given in Table 4.1.

	Linear systems	Sorting	Query
S-polynomial	C	C or Q	Q
Buchberger	C	C or Q	Q
F4	C (or Q in a limited manner)	C or Q	Q
F5	C	C or Q	Q
F5C	C (or Q in a limited manner)	C or Q	Q

Table 4.1: The possible choices of classical or quantum algorithms for the computation of Gröbner basis. In this table, each row takes one algorithm and access the choice between classical and quantum algorithm for sub-processes. The possible options are denoted by the symbols 'C' and 'Q'.

The followings are our conclusions of the research. The quantum query algorithm for finding the maximum (concerning the monomial order) is effective for detecting the head terms in polynomials. It is also useful to give the matrix representation of polynomials. As for the processes involving the reduction of polynomials, the choice of classical linear algorithm is inevitable, so long as the corresponding matrices are not of full-rank. In the F4 algorithm, the quantum algorithms are applicable only in the cases where we could solve the linear equation $Ax = b$ with the assurance that A is non-singular. In the F5 and F5C algorithms, the reduction to echelon in *top reduction* should be executed by Gauss-Jordan elimination. The F5C algorithm involves the process of turning the non-reduced Gröbner basis into the reduced one, to which the HHL algorithm is surely applicable.

In this article, we investigate the feasibility of quantum algorithms concerning Gröbner basis. However, some topics have not been touched yet. According to the reviews which the authors had given on other occasions, [KK19, Kik21], the symbolic computations for quantum physics inevitably require the computation of primary ideal decomposition, because it is another answer to the solution of the eigenvalue problem. Recall that the primary ideal decomposition could be done through the computation of Gröbner basis [Möl93, GTZ88, Laz92]. The primary ideal decomposition of polynomial ideals is to give the following representation to an ideal I

$$I = Q_1 \cap \cdots \cap Q_n$$

where $\{Q_i\}$ are primary ideals [Eis13]. This decomposition eases the root-finding of the set of polynomials that generate I : we have only to find the zeros for the decomposed

component $\{Q_i\}$, where the generating polynomials might be simpler. The primary ideal decomposition has the counterpart in number theory, namely, the prime factorization of integer n ,

$$n = p_1^{d_1} \cdots p_n^{d_n},$$

which is computable by Shor’s quantum algorithm. As for the applicability of the primary ideal decomposition to pragmatic studies, there is an example: the electronic structure of hydrogen molecules has been simulated from the first principles through the primary ideal decomposition of polynomial equations, which represent the Hartree-Fock equation [Kik13]. If quantum computing is fully introduced in the computation of Gröbner basis, we will arrive at a new era, where the problem of quantum physics could be pursued by the alliance of symbolic and numeric computations.

Maybe the effective computation of Gröbner basis is desired in another area, namely, in artificial intelligence. Automated geometric theorem proving can be done through the computation of Gröbner basis [KS86], and we could expect many applications in artificial intelligence. Nowadays the dominant trends in artificial intelligence are led by deep learnings, and the interest in theorem-proving seems to be eclipsed by them. The research of quantum algorithms is also pulled toward deep learnings. However, if the quantum algorithms are to improve the efficiency of symbolic computations, they shall facilitate the development of artificial intelligence which exploits theorem-proving on the rigorous foundation: this would be the advent of thinking machines.

References

- [ACM84a] Dennis S Arnon, George E Collins, and Scott McCallum. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM Journal on Computing*, 13(4):865–877, 1984.
- [ACM84b] Dennis S Arnon, George E Collins, and Scott McCallum. Cylindrical algebraic decomposition II: an adjacency algorithm for the plane. *SIAM Journal on Computing*, 13(4):878–889, 1984.
- [AK99] Ashish Ahuja and Sanjiv Kapoor. A quantum algorithm for finding the maximum. *arXiv preprint quant-ph/9911082*, 1999.
- [Buc65a] Bruno Buchberger. *An algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ideal*. PhD thesis, Ph. D. thesis, University of Innsbruck, Austria, 1965.
- [Buc65b] Bruno Buchberger. Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. *PhD thesis, Universitat Innsbruck*, 1965.
- [Buc76] Bruno Buchberger. A theoretical basis for the reduction of polynomials to canonical forms. *ACM SIGSAM Bulletin*, 10(3):19–29, 1976.

- [Buc79] Bruno Buchberger. A criterion for detecting unnecessary reductions in the construction of gröbner-bases. In *International Symposium on Symbolic and Algebraic Manipulation*, pages 3–21. Springer, 1979.
- [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.
- [Cop02] Don Coppersmith. An approximate fourier transform useful in quantum factoring. *arXiv preprint quant-ph/0201067*, 2002.
- [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.
- [DHM⁺18] Danial Dervovic, Mark Herbster, Peter Mountney, Simone Severini, Nairi Usher, and Leonard Wossnig. Quantum linear systems algorithms: a primer. *arXiv preprint arXiv:1802.08227*, 2018.
- [DJ92] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992.
- [DNDHG16] DIEP DO NGOC and NGUYEN VAN MINH DO HOANG GIANG. Quantum gauss-jordan elimination and simulation of accounting principles on quantum computers. *Preprint*, 2016. https://www.researchgate.net/publication/2196891_Quantum_Gauss_Jordan_Elimination.
- [DP13] Wolfram Decker and Gerhard Pfister. *A first course in computational algebraic geometry*. Cambridge University Press, 2013.
- [Dub90] Thomas W Dubé. The structure of polynomial ideals and gröbner bases. *SIAM Journal on Computing*, 19(4):750–773, 1990.
- [DW19] Ronald De Wolf. Quantum computing: Lecture notes. *arXiv preprint arXiv:1907.09415*, 2019.
- [EH11] Viviana Ene and Jürgen Herzog. *Gröbner Bases in Commutative Algebra*, volume 130. American Mathematical Soc., 2011.
- [Eis13] David Eisenbud. *Commutative Algebra: with a view toward algebraic geometry*, volume 150. Springer Science & Business Media, 2013.
- [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.
- [Fau02] 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.
- [FGGS98] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. A limit on the speed of quantum computation for insertion into an ordered list. *arXiv preprint quant-ph/9812057*, 1998.
- [FGLM93] Jean-Charles Faugere, Patrizia Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero-dimensional gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.
- [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.
- [GTZ88] Patrizia Gianni, Barry Trager, and Gail Zacharias. Gröbner bases and primary decomposition of polynomial ideals. *Journal of Symbolic Computation*, 6(2-3):149–167, 1988.
- [HHL09] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.
- [HNS02] Peter Høyer, Jan Neerbek, and Yaoyun Shi. Quantum complexities of ordered searching, sorting, and element distinctness. *Algorithmica*, 34(4):429–448, 2002.
- [Jor17] Stephen Jordan. Quantum Algorithm Zoo, 2017. <https://quantumalgorithmzoo.org/>.
- [Kik13] Akihito Kikuchi. An approach to first principles electronic structure computation by symbolic-numeric computation. *QScience Connect*, 2013:14, 2013. <http://dx.doi.org/10.5339/connect.2013.14>.
- [Kik21] Akihito Kikuchi. *Quantum Mechanics built on Algebraic Geometry: Emerging Physics through Symbolic Computation*. Eliva Press, 2021.
- [KK19] Akihito Kikuchi and Ichio Kikuchi. Computational algebraic geometry and quantum mechanics: An initiative toward post contemporary quantum chemistry. *Journal of Journal of Multidisciplinary Research and Reviews*, 1:47–79, 2019.
- [KS86] Bernhard Kutzler and Sabine Stifter. Automated geometry theorem proving using buchberger’s algorithm. In *Proceedings of the fifth ACM symposium on Symbolic and algebraic computation*, pages 209–214, 1986.
- [Laz92] Daniel Lazard. Solving zero-dimensional algebraic systems. *Journal of symbolic computation*, 13(2):117–131, 1992.

- [Möl93] H Michael Möller. On decomposing systems of polynomial equations with finitely many solutions. *Applicable Algebra in Engineering, Communication and Computing*, 4(4):217–230, 1993.
- [Pra14] Anupam Prakash. *Quantum algorithms for linear algebra and machine learning*. University of California, Berkeley, 2014.
- [SB06] Till Stegers and Johannes Buchmann. Faugere’s f5 algorithm revisited. *IACR Cryptol. ePrint Arch.*, 2006:404, 2006.
- [Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.
- [Sim97] Daniel R Simon. On the power of quantum computation. *SIAM journal on computing*, 26(5):1474–1483, 1997.
- [Tar98] Alfred Tarski. A decision method for elementary algebra and geometry. In *Quantifier elimination and cylindrical algebraic decomposition*, pages 24–84. Springer, 1998.