Testing Shift-Equivalence of Polynomials Using Quantum Machines

D. Grigoriev[1]
Department of Computer Science &
Department of Mathematics
Penn State University
University Park, PA 16802 USA
email: dima@cse.psu.edu

The polynomials $f, g \in F[X_1, \dots, X_n]$ are called shift-equivalent if there exists a shift $(\alpha_1, \dots, \alpha_n) \in F^n$ such that $f(X_1 + \alpha_1, \dots, X_n + \alpha_n) = g$. The algorithms in three different cases are designed which produce the set of all shift-equivalences of f, g in polynomial time, herewith in the case of a

1. zero-characteristics field F the designed algorithm is deterministic;
2. prime residue field $F = \mathbf{F}_p$ and a reduced polynomial f, i.e. $\deg_{X_i}(f) \le p-1$, $1 \le i \le n$, the algorithm is randomized;
3. finite field $F = \mathbf{F}_q$ of the characteristic 2 the algorithm is quantum. For an arbitrary finite field $\mathbf{F}_q$ a quantum machine is designed which computes the group of all shift-self-equivalences of f, i.e. $(\beta_1, \dots, \beta_n) \in \mathbf{F}_q^n$ such that $f(X_1 + \beta_1, \dots, X_n + \beta_n) = f$.

1 Introduction

In the paper we deal with the problem of testing, whether two given polynomials $f, g \in F[X_1, \dots, X_n]$ are shift-equivalent, i.e. there exists a shift $\alpha_1, \dots, \alpha_n$ such that $f(X_1+\alpha_1, \dots, X_n+\alpha_n) = g$. The issue of considering polynomials up to the shifts appeared already in the context of the interpolation of shifted-sparse polynomials (see [8, 10, 7]), namely, the polynomials which become sparse after a suitable shift.

We present the algorithms for computing the group $S_{f,f}$ of the shifts $(\beta_1, \dots, \beta_n)$ such that $f(X_1 + \beta_1, \dots, X_n + \beta_n) = f$ and for testing, whether the set $S_{f,g}$ of the shifts $(\alpha_1, \dots, \alpha_n)$ for which $f(X_1 + \alpha_1, \dots, X_n + \alpha_n) = g$ is nonempty (in the latter case $S_{f,g} = (\alpha_1, \dots, \alpha_n) + S_{f,f}$ and the algorithm yields a certain $(\alpha_1, \dots, \alpha_n) \in S_{f,g}$). The nature and the complexity of the algorithms drastically depends on the characteristic of the ground field F. For the characteristic zero we design in the section 2 a deterministic algorithm for testing shift-equivalence which has a polynomial time complexity, if the degree of f grows slower than n.

For the positive characteristic p when $F = \mathbf{F}_p$ is the field of residues mod p and the polynomial f is reduced, i.e. the degree with respect to each variable $\deg_{X_i}(f) \le p-1$, $1 \le i \le n$, we design in the section 3 a randomized algorithm for testing shift-equivalence which has a polynomial running-time, if p grows slower than a certain polynomial in n/d.

For an arbitrary finite ground field F and the degree of f we design in the section 4 a quantum machine (for this computational model and the background see e.g. [1, 13, 14, 16]) which computes the group $S_{f,f}$. Observe that the developed in the section 4 methods actually allow one to design a quantum machine which for a given action of an abelian group on a finite set, computes the stabilizator subgroup of a given element from the set (as the author recently learned, the problem of computing the stabilizator subgroup by a quantum machine was also solved in [17] with a better complexity bound). In [18] a quantum machine was constructed which allows one to test, whether a given function has a hidden linear structure, or to find the period of a periodic univariate function with small preimages (the latter result generalizes [13]). The method exhibited in the section 4 has a common point with [18] in applying the Fourier transform to the similar initial configurations (actually, this idea rises to [14]), but it is easier since it does not use the uniqueness of a hidden linear structure and estimations of the amplitudes as in [18], but rather exploites the duality of $S_{f,f}$ with its group of characters.

When the characteristic of F is 2 we design a quantum machine which computes $S_{f,g}$. Moreover, if the abelian group being a direct product of cyclic groups each of the order 2 acts on a finite set, one can design a quantum machine, which tests, whether two elements from the set lie in the same orbit of the action of the group. It seems to be an open question, whether one could solve the latter problem by a quantum machine over a finite field of an arbitrary characteristics. The designed quantum machines run in polynomial time, if p grows slower than a certain polynomial in the input size $\begin{pmatrix} n+d \\ d \end{pmatrix}$ (being the number of the coefficients of f).

Now we formulate the main result of the paper.

Theorem 1) Let $f, g \in \mathbf{Q}[X_1, \dots, X_n]$, $\deg(f), \deg(g) \le d$ and the bit-size of the coefficients of f, g be less than M. A (deterministic) algorithm is designed which finds a basis (over $\mathbf{C}$) $v_1, \dots, v_k \in \mathbf{Q}^n$ of the linear space $S_{f,f} \subset \mathbf{C}^n$ of all the shift-selfequivalences of f. Moreover, the algorithm tests, whether the set of all shift-equivalences $S_{f,g} \subset \mathbf{C}$ is nonempty and in the later case produces an element

[1]Supported by NSF grant CCR-9424358.

$(\alpha_1,\dots,\alpha_n)\in S_{f,g}\cap \mathbf{Q}^n$. The running time of the algorithm can be bounded by $(M(dn)^d)^{O(1)}$.

2) Let $f,g\in \mathbf{F}_p[X_1,\dots,X_n]$ for a prime p, the degrees $\deg(f)$, $\deg(g)\le d$ and $\deg_{X_i}(f)$, $\deg_{X_i}(g)\le p-1$, $1\le i\le n$. A randomized algorithm is designed which finds a basis over $\mathbf{F}_p$ of the linear space $S_{f,f}\subset \mathbf{F}_p^n$. Moreover, the algorithm tests, whether $S_{f,g}\neq\emptyset$ and in the latter case produces an element $(\alpha_1,\dots,\alpha_n)\in S_{f,g}$. The running time of the algorithm does not exceed $(p^d\begin{pmatrix} n+d\\ d\end{pmatrix})^{O(1)}$.

3) Let $f,g\in \mathbf{F}_{p^m}[X_1,\dots,X_n]$, the degrees $\deg(f),\deg(g)\le d$. A quantum machine is designed which finds a basis over $\mathbf{F}_p$ of $S_{f,f}\subset(\mathbf{F}_{p^m})^n$. Moreover, when the fields characteristics $p=2$, a quantum machine is designed which computes $S_{f,g}\subset(\mathbf{F}_{2^m})^n$ in the form similar to 2). The running times of the quantum machines are less than $(pm\begin{pmatrix} n+d\\ d\end{pmatrix})^{O(1)}$.

2 Testing shift-equivalence of polynomials over zero-characteristic field: deterministic algorithm

Let $f,g\in\mathbf{Q}[X_1,\dots,X_n]$ be two polynomials with $\deg(f)$, $\deg(g)\le d$ and with the size of rational coefficients less than M. Actually, one could consider the coefficients of f,g from a larger (say, algebraic number) field, but we stick with the rational coefficients just for simplifying the bounds on the size of the output data.

Denote by $S_{f,g}\subset\overline{\mathbf{Q}}^n$ (herewith the bar denotes the algebraic closure) the set of all shift-equivalence of f and g, i.e. $(\beta_1,\dots,\beta_n)\in\overline{\mathbf{Q}}^n$ such that $f(X_1+\beta_1,\dots,X_n+\beta_n)=g(X_1,\dots,X_n)$. If $S_{f,g}\neq\emptyset$ we say that f and g are shift-equivalent. In this section we design a deterministic algorithm which computes $S_{f,g}$. Observe that if $(\alpha_1,\dots,\alpha_n)\in S_{f,f}$ then for any integer m we have $(m\alpha_1,\dots,m\alpha_n)\in S_{f,f}$. Hence $(t\alpha_1,\dots,t\alpha_n)\in S_{f,f}$ holds for any $t\in\overline{\mathbf{Q}}$. Thus, considering t as a new variable, we get that

$$\begin{aligned}0 &= \frac{df(X_1+t\alpha_1,\dots,X_n+t\alpha_n)}{dt}\\ &= \left(\alpha_1\frac{\partial f}{\partial X_1}+\cdots+\alpha_n\frac{\partial f}{\partial X_n}\right)(X_1+t\alpha_1,\dots,X_n+t\alpha_n)\end{aligned}$$

Substituting $t=0$ in the latter identity, we obtain that $\alpha_1\frac{\partial f}{\partial X_1}+\cdots+\alpha_n\frac{\partial f}{\partial X_n}=0$. Inversing this arguing, we conclude that $S_{f,f}\subset\overline{\mathbf{Q}}^n$ is a linear subspace. Therefore, $S_{f,g}$ is a linear variety of the same dimension as $S_{f,f}$ (if $S_{f,g}$ is nonempty).

Observe that the variety $S_{f,g}$ is defined over $\mathbf{Q}$, therefore the subspace $S_{f,f}$ has a basis from $\mathbf{Q}^n$ (one could obtain it from the system of linear equations $\alpha_1\frac{\partial f}{\partial X_1}+\cdots+\alpha_n\frac{\partial f}{\partial X_n}=0$ in the variables $\alpha_1,\dots,\alpha_n$). Furthermore, $S_{f,g}$ contains a vector from $\mathbf{Q}^n$, indeed, take any vector $\beta\in S_{f,g}$ and all its conjugates $\beta=\delta_1(\beta),\delta_2(\beta),\dots,\delta_N(\beta)\in S_{f,g}$ over $\mathbf{Q}$, then $\frac{1}{N}\sum_{1\le j\le N}\delta_j(\beta)\in S_{f,g}\cap\mathbf{Q}^n$, thus $S_{f,g}$ is definable by a linear system over $\mathbf{Q}$.

For brevity denote $S^{(i)}=S_{\partial f/\partial X_i,\partial g/\partial X_i}$, $1\le i\le n$.

Lemma 1 $S_{f,g}=\cap_{1\le i\le n}S^{(i)}\cap\{(\alpha_1,\dots,\alpha_n)\in\overline{\mathbf{Q}}^n : f(\alpha_1,\dots,\alpha_n)=g(0,\dots,0)\}$.

Relying on lemma 1, the algorithm finds each $S^{(i)}$, $1\le i\le n$ by a linear over $\mathbf{Q}$ system defining $S^{(i)}$, using the recursion on the degree. Then the algorithm finds a linear over $\mathbf{Q}$ system defining the intersection $\cap_{1\le i\le n}S^{(i)}$ and substitutes the general (parametric) solution $(A_1,\dots,A_n)=v+\lambda_1v_1+\cdots+\lambda_kv_k$ of the latter system (here $\lambda_1\dots,\lambda_k$ are parameters, the vectors $v,v_1,\dots,v_k\in\mathbf{Q}^n$, $k=\dim\cap_{1\le i\le n}S^{(i)}$, and $v_1\dots,v_k$ are linearly independent) into f. Due to lemma 1 the set of the vectors $(A_1,\dots,A_n)$ satisfying the equation $f(A_1,\dots,A_n)=g(0,\dots,0)$, coincides with $S_{f,g}$.

Hence the equation $f(A_1,\dots,A_n)=g(0,\dots,0)$ determines a linear variety V in the space $\Lambda\simeq\overline{\mathbf{Q}}^k$ of parameters $(\lambda_1,\dots,\lambda_k)$. There could occur one of the following three cases. In the first case $V=\emptyset$, i.e. $S_{f,g}=\emptyset$, this means that the polynomial $f(A_1,\dots,A_n)-g(0,\dots,0)\in\mathbf{Q}[\lambda_1,\dots,\lambda_k]$ equals to a nonzero constant from $\mathbf{Q}$. In the second case $V=\Lambda$, i.e. $S_{f,g}=\cap_{1\le i\le n}S^{(i)}$, it is equivalent to identical vanishing of the polynomial $f(A_1,\dots,A_n)-g(0,\dots,0)$. In the last case V is a hyperplane in Λ, given by a linear equation $\sum_{1\le j\le k}c_j\lambda_j-c_0=0$ for suitable $c_j\in\mathbf{Q}$. Therefore,

$$f(A_1,\dots,A_n)-g(0,\dots,0)=c\quad\left(\sum_{1\le j\le k}c_j\lambda_j-c_0\right)^{\delta}$$

for an appropriate $c\in\mathbf{Q}$, where $\delta=\deg_{\lambda_1,\dots,\lambda_k}f(A_1,\dots,A_n)$. Let us find all c_j. Checking, whether the polynomial $f(A_1,\dots,A_n)-g(0,\dots,0)$ is homogeneous, we detect, whether $c_0=0$. If $c_0\neq0$ we set $c_0=1$ and for every $1\le j\le k$ replace λ_ℓ, for all $\ell\neq j$ by zeroes in $f(A_1,\dots,A_n)-g(0,\dots,0)$, as a result we obtain a univariate polynomial $\psi_j=f(A_1,\dots,A_n)-g(0,\dots,0)\big|_{\lambda_\ell=0,\ell\neq j}=c(c_j\lambda_j-1)^\delta$. The algorithm finds c_j calculating $GCD\left(\psi_j,\frac{d\psi_j}{d\lambda_j}\right)=c_j\lambda_j-1$. If on the opposite $c_0=0$, then for each pair $1\le j_1,j_2\le k$ we make a substitution $\lambda_{j_1}=1$ and $\lambda_\ell=0$ for all $\ell\neq j_1,j_2$, as a result the algorithm either finds the quotient c_{j_2}/c_{j_1} or returns $c_{j_1}=0$ similar to the situation $c_0=1$. This completes the description of the recursive algorithm which computes $S_{f,g}$.

In particular, this allows one to test, whether f and g are shift-equivalent.

Now we estimate the number of arithmetic operations in the described algorithm. The number of monomials in f,g and the number of taking the derivatives can be bounded by $\begin{pmatrix} d+n\\ d\end{pmatrix}^{O(1)}$. At each step of recursion for constructing the intersection $\cap_{1\le i\le n}S^{(i)}$ the algorithm solves a linear system in n variables, it requires $n^{O(1)}$ arithmetic operations. After that the calculating of the substitution $f(A_1,\dots,A_n)$ needs $\begin{pmatrix} d+n\\ d\end{pmatrix}^{O(1)}$ operations, and finally computing c_j takes $n^{O(1)}$ operations. Thus, the number of arithmetic operations of the algorithm does not exceed $\begin{pmatrix} d+n\\ d\end{pmatrix}^{O(1)}$, i.e. is polynomial in the input size.

Now we estimate the bit-size of the occurring intermediate coefficients. The bit-size of the coefficients of any involved partial derivative is less than $d(\log n)M$. Denote by M_ℓ, $0\le b\le d$ the bit-size of the coefficients of the linear systems representing S_{f_ℓ,g_ℓ} for intermediate in the recursion polynomials of degrees ℓ. Then at the current step of the recursion the size of the coefficients in a linear system representing $\cap_{1\le i\le n}S^{(i)}$ can be bounded by $n^{O(1)}M_\ell$, then the size of the coefficients c_j does not exceed $(nd)^{O(1)}M_\ell$ by the subresultant theorem [11]. Hence $M_{\ell+1} \leq (nd)^{O(1)}M_\ell$ and we conclude that $M_d \leq (nd)^{O(d)}M$ and the bit-size of all occurring coefficients is also less than $M(nd)^{O(d)}$. Therefore, the running time of the described algorithm does not exceed $(M(nd)^d)^{O(1)}$ which completes the proof of theorem 1). When $d = n^{o(1)}$ then $(nd)^{O(d)} \leq \begin{pmatrix} d+n \\ n \end{pmatrix}^{O(1)}$ and the bit complexity of the described algorithm is polynomial. When d grows faster than, say, n^2 it is more profitable for computing $S_{f,g}$ to solve a system of polynomial equations $f(X_1+\alpha_1,\dots,X_n+\alpha_n) = g(X_1,\dots,X_n)$ in n variables $\alpha_1,\dots,\alpha_n$ with the running time $(Md^{n^2})^{O(1)}$ [3].

3 Testing shift-equivalence of reduced polynomials over a prime residues field: randomized algorithm

Let the polynomials $f, g \in \mathbf{F}_p[X_1,\dots,X_n]$, $\deg(f), \deg(g) \leq d$, where p is a prime, be reduced, namely $\deg_{X_i}(f), \deg_{X_i}(g) \leq p-1$, $1 \leq i \leq n$. In this section we design a polynomial-time randomized algorithm which computes $S_{f,g} \subset \mathbf{F}_p^n$. Observe that $S_{f,f}$ is a linear subspace over $\mathbf{F}_p$ and $S_{f,g} = v + S_{f,f}$ for an arbitrary vector $v \in S_{f,g}$ (if $S_{f,g} \neq \emptyset$).

Notice that since f, g are reduced, lemma 1 from the section 2 holds for $S_{f,g}$ also in the case under consideration.

Let $q = p^m$, a polynomial $h \in \mathbf{F}_q[X_1,\dots,X_n]$. The following lemma 2 was told the author by R. Smolensky [15] and strengthens Schwartz's lemma [12] for finite fields. Observe that when $n \leq q \deg h$ and $\deg_{X_i}(h) \leq q-2$, $1 \leq i \leq n$, lemma 2 follows from [9] (for arbitrary h a weaker bound was proved in [5]).

Lemma 2 If h has a zero in $\mathbf{F}_q^n$ then h has at least $q^{n-\deg(h)}$ zeroes.

Now we describe a randomized algorithm which computes $S_{f,g} \subset \mathbf{F}_p^n$. Similar to the section 2 the algorithm by recursion on the degree computes $S^{(i)}$, $1 \leq i \leq n$ representing each $S^{(i)}$ by a linear system over $\mathbf{F}_p$. Then the algorithm produces a linear system which represents the intersection $\cap_{1\leq i\leq n} S^{(i)}$ yields the general (parametric) solution $(A_1,\dots,A_n) = v + \lambda_1 v_1 + \cdots + \lambda_k v_k$, where $v, v_1,\dots,v_k \in \mathbf{F}_p^n$, $k = \dim_{\mathbf{F}_p}\left(\cap_{1\leq i\leq n} S^{(i)}\right)$ of this linear system (cf. section 2). After that the algorithm substitutes $(A_1,\dots,A_n)$ in f. Due to lemma 1 $S_{f,g}$ is isomorphic to the set of the solutions of the polynomial over $\mathbf{F}_p$ equation $f(A_1,\dots,A_n) = g(0,\dots,0)$ in the parameters $\lambda_1,\dots,\lambda_k$, one could consider w.l.o.g. $S_{f,g}$ as a linear over $\mathbf{F}_p$ variety in the space $\Lambda \simeq \mathbf{F}_p^k$ of the parameters. Lemma 2 implies that $s = \dim_{\mathbf{F}_p} S_{f,g} \geq k - \deg f \geq k - d$ (if $S_{f,g} \neq \emptyset$).

In [8] it is proved that if in a set U to choose randomly independently N times elements $u \in U$ then the number y of the times when a chosen u belongs to a fixed subset $\emptyset \neq A \subset U$, satisfies with the probability greater than $1-\delta$ the following inequalities: $\frac{1}{2}\frac{\#A}{\#U} \leq \frac{y}{N} \leq \frac{3}{2}\frac{\#A}{\#U}$, where

$$N = \frac{\#U}{\#A} \cdot 16 \cdot \log_2(2/\delta).$$

The randomized algorithm under description for computing $S_{f,g}$ checks first, whether $0 \in S_{f,g}$, if yes then we set the vector $u_0 = 0 \in S_{f,g}$. If not then the algorithm chooses N times randomly independently elements from the set $U = \Lambda$, herewith $A = A_0 = S_{f,g}$ and $\delta = \left(n \begin{pmatrix} n+d \\ d \end{pmatrix}\right)^{-2}$. Hence $\frac{\#U}{\#A} \leq p^d$. Then with the probability greater than $1-\delta$ among the chosen $N = O(p^d d \log n)$ vectors there would be a vector $u_0 \in S_{f,g}$, (one could easily check the membership to $S_{f,g}$), provided that $S_{f,g} \neq \emptyset$. If none of the chosen N vectors belongs to $S_{f,g}$, the algorithm returns that $S_{f,g} = \emptyset$.

After that the algorithm makes $2N$ independent choices of the elements from U. Among them with the probability greater than $1-\delta$ there is a vector $u_1 \in S_{f,g}$ such that $u_1 - u_0 \neq 0$ (herewith we take $A = A_1 = S_{f,g} \backslash \{u_0\}$, obviously $\#A_1 \geq \frac{1}{2}\#A_0$). Thereupon making again $2N$ independent choices the algorithm with the probability greater than $1-\delta$ finds a vector $u_2 \in S_{f,g}$ such that the vectors $u_2 - u_0, u_1 - u_0$ are linearly independent. Herewith we take $A = A_2 = S_{f,g} \backslash \mathcal{L}\ \{u_0, u_1\}$ where $\mathcal{L}\ (u_0', \cdots, u_\ell')$ denotes the minimal linear variety which contains the points $u_0',\dots,u_\ell'$ (clearly $\mathcal{L}\ \{u_0,u_1\}$ is a line), obviously $\#A_2 \geq \frac{1}{2}\#A_0$. Continuing in this way, the algorithm makes at most $s \leq k$ rounds of $2N$ independent choices, while it is possible to find the vectors $u_0, u_1, \dots, u_{s_0} \in S_{f,g}$, $s_0 \leq s$ such that the differences $u_1 - u_0, \dots, u_{s_0} - u_0$ are linearly independent. The algorithm returns that $S_{f,g} = u_0 + \overline{\lambda}_1(u_1 - u_0) + \cdots + \overline{\lambda}_{s_0}(u_{s_0} - u_0)$, where $\overline{\lambda}_1,\dots,\overline{\lambda}_{s_0}$ are parameters from $\mathbf{F}_p$.

The algorithm finds $S_{f,g}$ correctly with the probability at least $(1-\delta)^{n\begin{pmatrix} n+d \\ d \end{pmatrix}} \geq 1 - \left(n \begin{pmatrix} n+d \\ d \end{pmatrix}\right)^{-1}$, because the algorithm calls recursively to itself at most $\begin{pmatrix} n+d \\ d \end{pmatrix}$ times since the number of nonvanishing partial derivatives of f does not exceed $\begin{pmatrix} n+d \\ d \end{pmatrix}$, and at each recursive step the algorithm makes at most n rounds of $2N$ independent choices as described above. Notice that if $S_{f,g} = \emptyset$, the algorithm always returns the correct answer.

Finally, estimate the running time of the algorithm. As already mentioned, there are at most $\begin{pmatrix} n+d \\ d \end{pmatrix}$ recursive calls of the algorithm to itself. At each recursive step the algorithm first finds (deterministically) the intersection $\cap_{1\leq i\leq n} S^{(i)}$, by means of solving a linear over $\mathbf{F}_p$ system with at most n variables, that requires $((\log p)n)^{O(1)}$ running time. Then the algorithm makes at most n rounds of choosing $2N$ vectors from $U = \Lambda$, it takes $\left(p^d \begin{pmatrix} n+d \\ d \end{pmatrix}\right)^{O(1)}$, which completes the proof of theorem 2).

Notice that the time bound of the algorithm is better than the time bound p^n of the trivial search in $\mathbf{F}_p^n$ when $d = o(n)$. In this case the time bound of the algorithm is polynomial in the input size $\log p \cdot \begin{pmatrix} n+d \\ d \end{pmatrix}$ when $p = \left(\frac{n}{d}\right)^{O(1)}$.

4 Testing shift-equivalence of polynomials over a finite field: quantum computation

Let $g = p^m$ and the polynomials $f, g \in \mathbf{F}_q[X_1,\dots,X_n]$, $\deg(f)$, $\deg(g) \leq d$. In this section we design a quantum machine which computes $S_{f,f} \subset \mathbf{F}_q^n$ and, furthermore, in the case of the fields characteristic $p = 2$ we design a quantum machine which computes $S_{f,g}$. Observe as above that $S_{f,f}$ is an abelian group and $S_{f,g} = v + S_{f,f}$ for an arbitrary $v \in S_{f,g}$ (if $S_{f,g} \neq \emptyset$).

The core of a quantum machine, a concept being an extension of a randomized algorithm (see e.g. [1, 16]), is a fast unitary transformation. In [13] it was shown that a quantum machine could compute in polynomial time the Fourier transform ϕ_n for the cyclic group Z_n of the order n for "smooth" n, namely $n = p_1 \cdots p_\ell$ being a product of pairwise distinct small primes. In [4] ϕ_{2^k} was computed by a quantum machine based on the fast Fourier transform. First we show (although we do not immediately use it below) that ϕ_{p^k} for any small p could be computed recursively on k by a quantum machine in a more succinct way using the product-formula for Fourier transform [2], which in its turn easily entails the fast Fourier transform algorithm.

The matrix $\phi_p = \frac{1}{\sqrt{p}}\left(\exp\left(\frac{2\pi i}{p} s\ell\right)\right)_{1\le s,\ell\le p}$ the quantum machine computes directly. For the recursive step, let w be a primitive root of unity of the degree p^{k+1}. Denote by D a square $p^k \times p^k$ diagonal matrix with the diagonal elements being successive powers of $w : 1, w, w^2, \ldots, w^{p^k-1}$. Denote by I_ℓ the unit $\ell \times \ell$ matrix. Then the following product-formula

$$\phi_{p^{k+1}} = \left(I_{p^k} \otimes \phi_p\right) \begin{pmatrix} I_{p^k} & & & & \bigcirc \\ & D & & & \\ & & D^2 & & \\ & & & \ddots & \\ \bigcirc & & & & D^{p-1} \end{pmatrix} \left(\phi_{p^k} \otimes I_p\right)$$

allows one to compute $\phi_{p^{k+1}}$ recursively by a quantum machine within time $O((kp)^2)$. Also observe that this gives a representation of $\phi_{p^{k+1}}$ as a product of $O(k)$ matrices, while [4] provides for it the product of $O(k^2)$ matrices.

Remark that as any finite abelian group G is a direct product $Z_{p_\ell^{k_1}} \times \cdots \times Z_{p_\ell^{k_\ell}}$ of the cyclic groups its Fourier transform $\phi_G = \phi_{p_1^{k_1}} \otimes \cdots \otimes \phi_{p_\ell^{k_\ell}} = \left(\phi_{p_1^{k_1}} \otimes I_{p_2^{k_2}} \otimes \cdots \otimes I_{p_\ell^{k_\ell}}\right)$ $\left(I_{p_1^{k_1}} \otimes \phi_{p_2^{k_2}} \otimes \cdots \otimes I_{p_\ell^{k_\ell}}\right) \cdots \left(I_{p_1^{k_1}} \otimes I_{p_2^{k_2}} \otimes \cdots \otimes \phi_{p_\ell^{k_\ell}}\right)$ could be computed by a quantum machine within time $O\left(\sum_{1\le i\le \ell} (p_i k_i)^2\right)$.

First we design a quantum machine which computes the group $S_{f,f} \subset \mathbf{F}_q^n$. This construction extends essentially the idea from [14]. We utilize the notations and terminology from the quantum computations which one could find in [1, 13, 14, 16]. Actually, the described algorithm and the above quantum computation of ϕ_G allows one to solve the following problem by means of a quantum machine. Let a finite abelian group G with all the primes dividing its order, being small, act on a set. The algorithm enables one to find for each element of the set the subgroup of G which preserves this element (the stabilizator subgroup, see also [17]). Furthermore, if G is a direct product of cyclic groups each of the order 2, one can design a quantum machine which for any pair of elements of the set tests, whether these two elements are on the same orbit of the action of G. In the case under consideration $G = Z_p \times \cdots \times Z_p$ is the direct product of mn copies of Z_p, herewith the action of $(Z_p)^m$ on each variable X_i, $1 \le i \le n$ is isomorphic to the action of the additive group of $\mathbf{F}_q$ by the shifts.

The quantum machine under description starts with the initial configuration (cf. [1, 13, 14, 16, 18])

$$C = \frac{1}{\left(\sqrt{q}\right)^n} \sum_{(\alpha_1,\ldots,\alpha_n)\in \mathbf{F}_q^n} \left|\alpha_1, \ldots, \alpha_n, f(X_1+\alpha_1, \ldots, X_n+\alpha_n)\right\rangle,$$

i.e. each basic state $\left|\alpha_1, \ldots, \alpha_n, f(X_1+\alpha_1, \ldots, X_n+\alpha_n)\right\rangle$ is taken with the amplitude $\frac{1}{(\sqrt{q})^n}$. Notice that each basic state is a basic ort in $\left(q^n \cdot q^{\binom{n+d}{d}}\right)$-dimensional $\mathbf{C}$-space with the Hermitean metric. Let $w^{(1)}, \ldots, w^{(m)} \in \mathbf{F}_q$ be a basis over $\mathbf{F}_p$. Then one can represent each basic state $|\alpha_1, \ldots, \alpha_n, f(X_1+\alpha_1, \ldots, X_n+\alpha_n)\rangle$ in the form $|\alpha_1^{(1)}, \ldots, \alpha_1^{(m)}, \ldots, \alpha_n^{(1)}, \ldots, \alpha_n^{(m)}, f(X_1+\alpha_1, \ldots, X_n+\alpha_n)\rangle$ where $\alpha_\ell = \sum_{1\le j\le m} \alpha_\ell^{(j)} w^{(j)}$, $\alpha_\ell^{(j)} \in \mathbf{F}_p$, $1 \le \ell \le n$. The additive group of $\mathbf{F}_q^n$ acts on the first nm components as a direct product $(Z_p)^{nm}$.

Denote $Q = q^{\binom{n+d}{d}}$. The quantum machine applies to C the matrix (see above) $\phi_p \otimes \cdots \otimes \phi_p \otimes I_Q$ where the tensor product of ϕ_p is taken nm times (cf. [18]). Then in the resulting configuration any basic state $|\chi_1, \ldots, \chi_{nm}, \overline{f}\rangle$ where $\chi_\ell : \mathbf{Z}/p\mathbf{Z} \to \mathbf{C}$, $1 \le \ell \le nm$ are the characters of the cyclic additive group of $\mathbf{F}_p$, i.e. $\chi_\ell(a) = \exp\left(\frac{2\pi i\, a\, b}{p}\right)$ for a suitable b and $\overline{f} = f(X_1+\beta_1, \ldots, X_n+\beta_n) \in \mathbf{F}_q[X_1, \ldots, X_n]$ for some $(\beta_1, \ldots, \beta_n) \in \mathbf{F}_q^n$, $\beta_\ell = \sum_{1\le j\le m} \beta_\ell^{(j)} w^{(j)}$ $1 \le \ell \le n$, occurs with the amplitude (cf. [13, 14, 18])

$$\begin{aligned} & \frac{1}{q^n} \sum_{(\sum \alpha_1^{(j)} w^{(j)}, \ldots, \sum \alpha_n^{(j)} w^{(j)}) \in S_{f,f}} \chi_1(\alpha_1^{(1)} \\ + \; & \beta_1^{(1)}) \cdots \chi_m(\alpha_1^{(m)} + \beta_1^{(m)}) \cdots \chi_{nm-m+1} \\ & (\alpha_n^{(1)} + \beta_n^{(1)}) \cdots \chi_{nm}(\alpha_n^{(m)} + \beta_n^{(m)}) \\ = \; & \frac{1}{q^n} \chi_1(\beta_1^{(1)}) \cdots \chi_{nm}(\beta_n^{(m)}) \sum_{(\sum \alpha_1^{(j)} w^{(j)}, \ldots, \sum \alpha_n^{(j)} w^{(j)}) \in S_{f,f}} \\ & \chi_1(\alpha_1^{(1)}) \cdots \chi_{nm}(\alpha_n^{(m)}) \end{aligned}$$

For every restriction of the character

$$\chi = \chi_1 \otimes \cdots \otimes \chi_{nm}\big|_{S_{f,f}} \text{ the sum } \sum_{\alpha \in S_{f,f}} \chi(\alpha) = \begin{cases} 0 & \text{if } \chi \not\equiv 1 \\ \#S_{f,f} & \text{if } \chi \equiv 1 \end{cases}$$

where $\alpha = (\sum \alpha_1^{(j)} w^{(j)}, \ldots, \sum \alpha_n^{(j)} w^{(j)})$. Thus, each of the basic states $|\chi_1, \ldots, \chi_{nm}, \overline{f}\rangle$ for which $\chi_1 \otimes \cdots \otimes \chi_{nm}\big|_{S_{f,f}} \equiv 1$ (and only these basic states) occurs in the resulting configuration with the same for each of them probability (which equals to the square of the absolute value of the amplitude, see [1, 13, 14, 16, 18]) $\frac{(\#S_{f,f})^2}{q^{2n}}$. Hence, each vector $(\chi_1, \ldots, \chi_{nm})$ such that $\chi_1 \otimes \cdots \otimes \chi_{nm}\big|_{S_{f,f}} \equiv 1$, occurs as the first nm coordinates of the basic states in the resulting configuration with the same for each of them probability $\frac{\# S_{f,f}}{q^n}$, because for the rest of Q coordinates there are $\frac{q^n}{\# S_{f,f}}$ possibilities for $\overline{f}$, each of them appearing with the same probability.

Since $S_{f,f}$ is an abelian subgroup of the additive group of $(\mathbf{F}_q)^n$, the order $\# S_{f,f} = p^k$ for a certain $0 \leq k \leq nm$. All the vectors of the characters $(\chi_1, \dots, \chi_{nm})$ such that the restriction $\chi_1 \otimes \cdots \otimes \chi_{nm} \Big|_{S_{f,f}} \equiv 1$ constitute the (multiplicative) group $\mathcal{S}$ being isomorphic to the vector space $(\mathbf{F}_p)^{nm-k}$ over $\mathbf{F}_p$.

Applying nm times independently the described quantum machine and each time observing the projection onto the first nm coordinates of a basic state of the resulting configuration, we obtain a sequence of nm elements from $\mathcal{S}$. The probability that the first $nm-k$ vectors (one can assume that they are chosen independently as each of them appears with the same probability, see above) among them form a basis of $\mathcal{S}$ over $\mathbf{F}_p$ is greater or equal to $(1-p^{-1})(1-p^{-2})(1-p^{-3})\cdots > \frac{1}{4}$.

Therefore, making 4 rounds each consisting of nm described applications of the quantum machine, with the probability greater than $1-(1-\frac{1}{4})^4 > \frac{2}{3}$, the quantum algorithm yields at one of the rounds a basis for the space $\mathcal{S}$ over $\mathbf{F}_p$. The algorithm returns as a basis the maximal set of linearly independent over $\mathbf{F}_p$ elements of $\mathcal{S}$ obtained at one of 4 rounds.

Having a basis of $\mathcal{S}$, the algorithm can uniquely select the subgroup $S_{f,f}$. Indeed, for every element $(\chi_1, \dots, \chi_{nm})$ from the yielded basis let $\chi_t(\alpha) = \exp\left(\frac{2\pi i\, \ell_t \alpha}{p}\right)$, $1 \leq t \leq nm$ for appropriate $0 \leq \ell_t < p$, then for any element $\left(\sum_{1 \leq j \leq m} \alpha_1^{(j)} w^{(j)}, \dots, \sum_{1 \leq j \leq m} \alpha_n^{(j)} w^{(j)}\right) \in S_{f,f}$ we have $\chi_1(\alpha_1^{(1)}) \cdots \chi_{nm}(\alpha_n^{(m)}) = 1$, i.e. $p \Big| \ell_1 \alpha_1^{(1)} + \cdots + \ell_{nm} \alpha_n^{(m)}$. Conversely, if the latter divisibility holds for every element from the basis then $\left(\sum_{1 \leq j \leq m} \alpha_1^{(j)} w^{(j)}, \dots, \sum_{1 \leq j \leq m} \alpha_n^{(j)} w^{(j)}\right) \in S_{f,f}$. These divisibility conditions constitute a (homogeneous) linear system over $\mathbf{F}_p$. Producing a basis of this linear system, the algorithm produces thereby a basis of $S_{f,f}$. This completes the description of the algorithm which computes $S_{f,f}$.

Now in the case of the fields characteristic $p = 2$ we design a quantum machine which tests, whether $S_{f,g} \neq \emptyset$, and if it is the case the machine yields an element $v \in S_{f,g}$. Together with the described above construction of $S_{f,f}$ this computes $S_{f,g} = v + S_{f,f}$. First the machine checks, whether $f \equiv g$, and if it is the case we are done by the above construction of $S_{f,f}$, so we can suppose w.l.o.g. that $f \not\equiv g$. Then applying the described above construction, the machine computes the groups $S_{f,f}$ and $S_{g,g}$. If $S_{f,f} \neq S_{g,g}$ then $S_{f,g} = \emptyset$. So we can assume that $S_{f,f} = S_{g,g}$.

Observe that $S = S_{f,f} \cup S_{f,g}$ is a group since $p = 2$. Notice also that S coincides with the group of all the shifts $(\alpha_1, \dots, \alpha_n) \in \mathbf{F}_q^n$ which preserve the unordered pair of the polynomials $\{f(X_1, \dots, X_n), g(X_1, \dots, X_n)\} = \{f(X_1 + \alpha_1, \dots, X_n + \alpha_n), g(X_1 + \alpha_1, \dots, X_n + \alpha_n)\}$.

To compute S the quantum machine as the basic states takes

$$|\alpha_1, \dots, \alpha_n, \{f(X_1 + \alpha_1, \dots, X_n + \alpha_n), g \\ (X_1 + \alpha_1, \dots, X_n + \alpha_n)\}\rangle$$

where $(\alpha_1, \dots, \alpha_n) \in \mathbf{F}_q^n$. Thus, a basic state could be treated as an ort from $\mathbf{C}$-space of the dimension $q^n \cdot \mathbf{a}$, where $\mathbf{a} = \frac{Q(Q+1)}{2}$. As in the above construction, the quantum machine applies the Fourier transform $\phi = \phi_2 \otimes \cdots \otimes \phi_2$ (nm times) to the first n coordinates, formally the machine multiplies the initial configuration

$$\frac{1}{(\sqrt{q})^n} \sum_{\alpha_1, \dots, \alpha_n \in \mathbf{F}_q} |\alpha_1, \dots, \alpha_n, \{f(X_1 + \alpha_1, \dots, X_n + \alpha_n), \\ g(X_1 + \alpha_1, \dots, X_n + \alpha_n\}\rangle$$

by the matrix $\phi \otimes I_{\mathbf{a}}$. Then as above the quantum machine computes the group S (by means of its basis over $\mathbf{F}_2$).

Obviously, $S_{f,g} \neq \emptyset \iff S_{f,f} \neq S$, and in this case we can take as v any element of the basis of S which does not belong to $S_{f,f}$. This completes the description of the quantum machine which computes $S_{f,g}$.

Finally, we estimate the complexity of the designed quantum machines. In the course of computing $S_{f,f}$ the machine computes (deterministically) for any $(\alpha_1, \dots, \alpha_n) \in \mathbf{F}_q^n$ the coefficients of the polynomial $f(X_1 + \alpha_1, \dots, X_n + \alpha_n)$ which requires $\left(m \log p \begin{pmatrix} n+d \\ d \end{pmatrix}\right)^{O(1)}$ time. Producing Fourier transform ϕ_p takes $p^{O(1)}$ time. So, the application of the Fourier transform runs in $\left(m\, p \begin{pmatrix} n+d \\ d \end{pmatrix}\right)^{O(1)}$ time. The machine makes $O(nm)$ such rounds and at the end solves (deterministically) a linear over $\mathbf{F}_p$ system of the size $O(nm)$. Thus, the running time of the designed quantum machine does not exceed $\left(m\, p \begin{pmatrix} n+d \\ d \end{pmatrix}\right)^{O(1)}$. The similar bound is valid for the quantum machine which computes $S_{f,g}$, this completes the proof of theorem 3.

Notice that this bound is always not worse that the complexity bound for the randomized algorithm designed in the section 3 (for $m = 1$). When p grows like $\begin{pmatrix} n+d \\ d \end{pmatrix}^{O(1)}$ the running time of the designed quantum machine is polynomial which is not the case for the randomized algorithm from the section 3.

Acknowledgements. The author is grateful to Aliosha Kitaev for valuable remarks.

References

[1] E. Bernstein, U. Vazirani, Quantum complexity theory, Proc. STOC, (ACM, 1993), 11–20

[2] T. Beth, Verfahren der schnellen Fourier-transformation, (Teubner, Stuttgart, 1984)

[3] A. Chistov, D. Grigoriev, Solving algebraic systems in subexponential time. I, II., Preprints LOMI E-9-83, E-10-83, Leningrad, 1983

[4] D. Coppersmith An approximate Fourier transform useful in quantum factoring Research report 19642, IBM, 1994

[5] D. Grigoriev, M. Karpinski, An approximate algorithm for the number of zeroes of arbitrary polynomials over $GF[q]$, Proc. FOCS (IEEE, 1991), 662–669

[6] D. Grigoriev, M. Karpinski, A zero-test and a interpolation algorithm for the shifted sparse polynomials, Proc. AEECC 1993, Lect. Notes in Comput. Sci., vol. 673, (Springer, Berlin, 1993) 162–169

[7] D. Grigoriev, Y. Lakshman, Algorithms for computing sparse shifts for multivariate polynomials, Proc. Intern. Symp. on Symbol. Algebr. Comput. (ACM, Montreal, 1995) 96–103

[8] R. Karp, M. Luby, N. Madras, Monte-Carlo approximation algorithms for enumeration problems, J. of Algorithms, 10, N3 (1989), 429–448

[9] M. Karpinski, I. Shparlinski, Efficient approximation algorithms for sparse polynomials over finite fields, Technical Report 94–029, ICSI, Berkeley, 1994

[10] Y. Lakshman, D. Saunders, On computing sparse shifts for univariate polynomials, Proc. Intern. Symp. on Symbol. Algebr. Comput. (ACM, Oxford, 1994)

[11] R. Loos, Generalized polynomial remainder sequences, in B. Buchberger, J. Calmet, R. Loos, eds., Computer Algebra, (Springer, Berlin, 1982)

[12] T. Schwartz, Fast probabilistic algorithms for verification of polynomial identities, J. ACM, 27 (1980), 701–717

[13] P. W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, Proc. FOCS, (IEEE, 1994), 124–134

[14] D. R. Simon, On the power of quantum computation, Proc. FOCS (IEEE, 1994), 116–123

[15] R. Smolensky, Private communication, 1995

[16] A. Yao, Quantum circuit complexity, Proc. FOCS (IEEE, 1993), 352–360

[17] A. Yu. Kitaev, Quantum measurements and the Abelian stabilizer problem, Preprint of the Institute for Theoretical Physics, Moscow, October 1995

[18] D. Boneh, R.Lipton, Quantum cryptanalysis of hidden linear functions, Lect. Notes Comput. Sci., 963 (1995), 424–437