Quantum Algorithm for Computing Distances Between Subspaces

Nhat A. Nghiem[1]

[1]*Department of Physics and Astronomy, State University of New York at Stony Brook, Stony Brook, NY 11794-3800, USA*

Geometry and topology have generated impacts far beyond their pure mathematical primitive, providing a solid foundation for many applicable tools. Typically, real-world data are represented as vectors, forming a linear subspace for a given data collection. Computing distances between different subspaces is generally a computationally challenging problem with both theoretical and applicable consequences, as, for example, the results can be used to classify data from different categories. Fueled by the fast-growing development of quantum algorithms, we consider such problems in the quantum context and provide a quantum algorithm for estimating two kinds of distance: Grassmann distance and ellipsoid distance. Under appropriate assumptions and conditions, the speedup of our quantum algorithm is exponential with respect to both the dimension of the given data and the number of data points. Some extensions regarding estimating different kinds of distance are then discussed as a corollary of our main quantum algorithmic method.

I. INTRODUCTION

Quantum computation has opened up a completely new frontier in computational science. A vast amount of difficult computational problems have been theoretically shown to be accelerated by quantum computation. Famous examples include integer factorization [1], unstructured database search [2], quantum simulation [3], the task of probing properties of a black-box function [4], solving linear system [5, 6], and approximating topological invariants [7]. More recently, the interplay between quantum science and machine learning, so-called quantum machine learning, has led to many fascinating works, such as quantum neural network [8, 9], quantum convolutional neural network [10], quantum support vector machine [11], etc. Under certain assumptions regarding input access, the exponential speedup is achievable, such as performing supervised learning and unsupervised learning using quantum processors [12] and performing fitting over large data set [13]. Unconditional proof of quantum advantage was provided in [14], where the authors showed that shallow circuits can completely outperform their classical counterparts.

Aside from the aforementioned instances, where the domain of investigation ranges from algebraic problems to data science & machine learning, the potential advantage of quantum computers has also been explored in (computational) topology & geometry domain. Lloyd et al. [15] provided a quantum algorithm, the so-called LGZ algorithm, for computing Betti numbers of simplicial complexes, a classic problem arising from topological data analysis. Many followed-up works, such as [16–19] have improved the running time, as well as implementation costs of the LGZ algorithm. In [20], the authors outlined a quantum algorithm for problems in computational geometry, showing significant speedup. In [21], a single-query, constant-time quantum algorithm is provided for detecting the homology class of closed curves, which is an interesting problem in computational topology. These examples suggest a potentially fruitful domain where quantum advantage could be further explored, since topology and geometry, while being a classical subject within the pure mathematical domain, have provided a solid foundation for many applications. For example, the field of computational conformal geometry [22], which encompasses modern differential geometry, Riemann geometry, and algebraic topology, has provided many useful tools for challenging problems in computer vision, medical imaging, such as surface classification, registration, etc.

Motivated by these developments, particularly regarding geometry & topology, we tackle the following problem that arises from the same context: computing distance between linear subspaces [23]. This problem enjoys great applications in machine learning and computer vision, etc., but is quite challenging from a computational point of view since it requires the ability to evaluate singular values of possibly large matrices. We will show that quantum algorithmic techniques could enhance such tasks, yielding significant speedup compared to standard classical methods under certain conditions. Our work thus contributes as another example where quantum algorithms could be beneficial for practical problems.

The structure of the paper is as follows. In Section II, we begin with some introduction to topology and geometry so as to review some preliminaries and build up intuition that is very beneficial to understand the framework behind two problems that we wish to solve, which are computing the distance between *linear subspaces* and between *ellipsoids*. The formal description and the classical solution to those problems are also presented accordingly. In Section III, we introduce some necessary recipes and tools that are crucial in our subsequent construction. Section IV is dedicated to our main result, which is an efficient quantum algorithm for computing the linear subspace distance, as well as giving

details on error analysis and the final statement regarding its running time. The quantum solution to the second problem—computing the distance between ellipsoids is presented in Section V, which is more or less an adaptive version of the method outlined in Section IV and a result combined with some well-known quantum algorithms, such as inverting a dense matrix [24]. We then show that how the main algorithms presented in section VI can be extended to estimate other kinds of distance between subspaces. In section VII, we showcase how Grassmann distance and ellipsoid distance are estimated in the so-called memory model, which is a special type of quantum data structure that allows more subtle loading of classical data. We then conclude with some comments on our results in Sec. VIII and discuss future prospects.

II. DISTANCE BETWEEN SUBSPACES

Intuitively, geometry and topology typically begin with a *set of points* and other sets built from them, resulting in the so-called space. The most familiar space is probably the Euclidean space of a given dimension n, denoted as R^n, where each 'point' is associated with an n-tuple $(x_1, x_2, ..., x_n)$ (also called coordinates), where each $x_i \in R$. The collection of such points, endowed with addition and scalar multiplication operations, forms a *linear vector space*. These concepts belong to a subject called linear algebra, which has found countless applications in science & engineering, due to its simplicity and versatility. A remarkable property of such Euclidean space is that the *distance* between two points can be defined as some function of their coordinates. The Euclidean space is an instance of a more general concept called *manifold*, with the distance between two points now generalized as the *geodesic distance*.
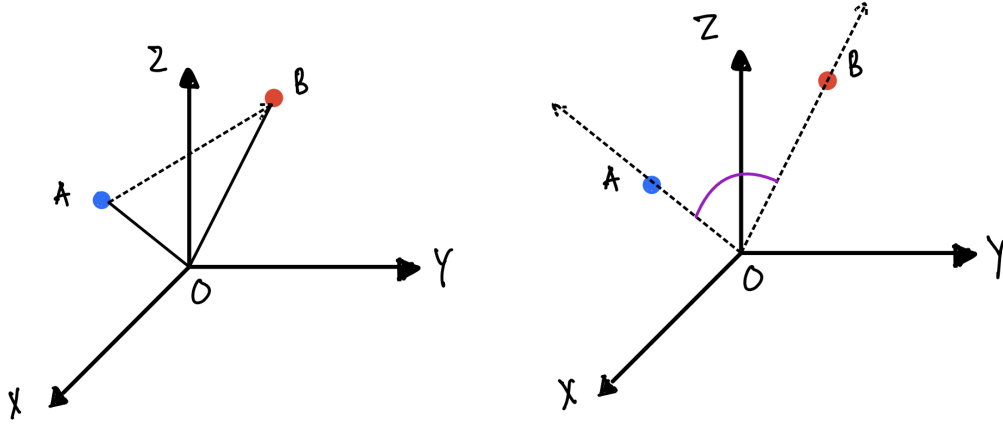

FIG. 1: **Left figure:** An elementary example of Euclidean space $\mathbb{R}^3$ with two points A and B (and original point O). The distance between A and B can be defined simply based on corresponding coordinates of A and B. **Right figure:** Illustrative example of distance between two subspaces. Instead of viewing A and B as two points, we consider two 1-dimensional subspaces each spanned by $\vec{OA}$ and $\vec{OB}$. Intuitively, the angle between $\vec{OA}$ and $\vec{OB}$ can be used as a measure of 'distance' between two 1-dimensional subspace.

Linear Subspaces: Given a vector space R^n, let $\{m_1, m_2, ..., m_k\}$ and $\{n_1, n_2, ..., n_k\}$ be two collections of linearly independent vectors. Without loss of generalization, we can assume them to be orthonormal sets. Denote M_A and M_B as two k-dimensional vector subspaces spanned by $\{m_1, m_2, ..., m_k\}$ and $\{n_1, n_2, ..., n_k\}$, respectively. Our main problem to compute the separation between M_A and M_B. As suggested in [23], M_A and M_B could be regarded as two elements, or two points, of the so-called Grassmannian manifold, $\text{Gr}(k,n)$. The separation between M_A and M_B could be computed as geodesic distance d_{M_A, M_B} between two points on manifold $\text{Gr}(k,n)$. Following [23], we now describe the approach for computing d_{M_A, M_B}.

First, we organize $\{m\} := \{m_1, m_2, ..., m_k\}$ and $\{n\} := \{n_1, n_2, ..., n_k\}$ as two column matrices $M, N \in R^{n \times k}$ (which means that the i-th column of M, N are m_i, n_i respectively). We now perform singular value decomposition (SVD) of $M^T N$:

$$M^T N = U \cdot \Sigma \cdot V^T, \tag{1}$$

with $\sigma \in R^{k \times k}$ having diagonal entries $\{\sigma_i\}_{i=1}^k$ as singular values of $M^T N$. The so-called Grassmann distance between

M_A and M_B can be computed as:

$$d_{M_A,M_B} = \sqrt{\sum_{i=1}^{k} \theta_i^2}, \tag{2}$$

with $\cos(\theta_i) = \sigma_i$. We remark that since $\{m\}$ and $\{n\}$ are orthonormal bases, then $0 \leq \sigma_i \leq 1$ for all i.

It is clear that the computational expense lies mostly in this SVD step. The running time of a classical algorithm that can compute the above Grassmann distance is obviously polynomial in n and k. The key steps include multiplying M^T to N and performing SVD on $M^T N$. The multiplication of $k \times n$ matrix M^T and $n \times k$ matrix N takes time $\mathcal{O}(n^2 k)$, meanwhile performing SVD on a $k \times k$ matrix using the so-called Jacobi rotations [25] takes $\mathcal{O}(k^3)$, yielding the total running time $\mathcal{O}(n^2 k + k^3)$. There are several works that provide quantum algorithms for performing SVD, such as [26, 27]. While there is a certain overlap, our subsequent quantum procedure employs somewhat different techniques since the data input to our problem is different, and in particular, our objective is also different.

Ellipsoids: Given a real symmetric positive definite matrix $M \in R^{n \times n}$, an ellipsoid $\mathcal{E}_A$ is defined as

$$\mathcal{E}_A = \{x \in R^n : x^T M x \leq 1\}. \tag{3}$$

The problem is that we need to find the separation, or distance, between two ellipsoids $\mathcal{E}_M$ and $\mathcal{E}_N$. A simple way to compute it is based on the **metric** defined on the cone of real symmetric positive definite matrices [23], which yields distance $\delta_{M,N}$ between $\mathcal{E}_M$ and $\mathcal{E}_N$:

$$\delta_{M,N} = \sqrt{\sum_{i=1}^{n} \log^2(\lambda_i(M^{-1}N))}, \tag{4}$$

where λ_i is the i-th eigenvalue of $A^{-1}B$. Since both A and B are real positive definite, $\{\lambda_i\}_{i=1}^{n}$ are real and positive. The running time of a classical algorithm that computes the above distance is similar to that of Grassmann distance, which is $\mathcal{O}(n^3)$, as they share similar computational steps, and in this case, we have that $k = n$.

III. SOME PRELIMINARIES

The previous section has provided some introduction and classical methods to compute Grassmann and ellipsoid distance. This section provides a review of necessary quantum tools that we would later use to develop our quantum algorithms.

Definition 1 (Block Encoding Unitary) *Let A be some Hermitian matrix of size $N \times N$. Let a unitary U have the following form:*

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix}.$$

Then U is said to be a block encoding of matrix A. Equivalently, we can write:

$$U = |\mathbf{0}\rangle \langle \mathbf{0}| \otimes A + \cdots$$

where $|\mathbf{0}\rangle$ denotes the first computational basis state in some larger Hilbert space. It is quite obvious from the above definition that

$$A_{ij} = (\langle \mathbf{0}| \otimes \langle i|)U(|\mathbf{0}\rangle \otimes |j\rangle), \tag{5}$$

where A_{ij} refers to the entry of A at i-th row and j-th column.

The next one is a very powerful technique that is often dubbed as quantum signal processing [28, 29].

Lemma 1 (Quantum Signal Processing [28]) *Let f be a polynomial of degree d with $|f(x)| \leq 1/2$ for all $x \in [-1, 1]$. Let U be a block encoding of some Hermitian matrix A. Then the following transformation*

$$\begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} \longrightarrow \begin{pmatrix} f(A) & \cdot \\ \cdot & \cdot \end{pmatrix}$$

is the so-called block encoding of $f(A)$ and can be realized using d applications of U and U^T plus one controlled-U gate.

The above Lemma is the most general statement of quantum signal processing technique, which shows flexibility in how the transformation of given matrix could be done. As being pointed out in [29], by making use of Jacobi-Anger expansion, one can efficiently transform the Hamiltonian operator H into a high-precision approximation operator of $\exp(-iHt)$. The scaling cost turned out to be optimal, which demonstrate the surprising power of such quantum signal processing technique.

Before proceeding further, we make the following remark. At the core of quantum signal processing technique is the operation on the block encoding (see Lemma 1 operator. The transformed operator is apparently block encoded. If we wish to apply this operator to some state and extra the measurement outcome from the output state, how can it be done in the subspace of the encoded operator? The answer is simple, as we do not need to worry about it. Let U denotes the unitary block encoding of some operator A. Let $|\mathbf{0}\rangle |\Phi\rangle$ denotes some state sharing the same dimension as U, where $|\Phi\rangle$ has same dimension as A, and obviously $|\mathbf{0}\rangle$ accounts for the remaining dimension. As we shall also show later, if we apply U to $|\mathbf{0}\rangle |\Phi\rangle$, then we have:

$$U |\mathbf{0}\rangle |\Phi\rangle = |\mathbf{0}\rangle A |\Phi\rangle + \sum_{j \neq 0} |j\rangle |Garbage_j\rangle \tag{6}$$

If we perform measurement on the above state and conditioning on the extra register being $|\mathbf{0}\rangle$, then the resulting state corresponds to the measurement from $A |\Phi\rangle$, which is of our interests. Therefore, the effect of high-dimension of the encoding step can be trivially removed by simply conditioning on $|\mathbf{0}\rangle$ subspace. Throughout the work, we would simply work in the subspace of encoded operator only, for simplicity.

The last recipe we need to construct our quantum algorithm is called an efficient matrix application, from a simple adaptive version of quantum random walk method [6] and is stated in the following lemma.

Lemma 2 (Efficient Matrix Application) *Given coherent oracle access to some s-sparse, Hermitian matrix H of dimension $n \times n$, and a given $n \times 1$ state $|b\rangle$. Then there is a unitary U_H that acts in the following way,*

$$U_H |0^m\rangle |b\rangle = |0^m\rangle (H/s) |b\rangle + |\Phi_\perp\rangle,$$

where $|\Phi_\perp\rangle$ is some unimportant state (not properly normalized) that is orthogonal to $|0^m\rangle (H/s) |b\rangle$, i.e, $|0^m\rangle \langle 0^m| \otimes \mathbf{1} |\Phi_\perp\rangle = 0$. The unitary U_H runs in time

$$\mathcal{O}\Big(\log(n), poly\big(\log(\frac{1}{\epsilon})\big)\Big),$$

where ϵ is the error tolerance.

Since the above prior results were collected for our purpose, we refer the readers to their respective original works for more thorough treatment. Now, we proceed to apply them to construct our main quantum algorithms.

IV. QUANTUM ALGORITHM FOR LINEAR SUBSPACES DISTANCE

This section is dedicated to the first problem that we described in a previous section: computing the Grassmann distance between linear subspaces, each spanned by an orthonormal set. We sketch the main idea behind our quantum algorithm as follows.

First, we remind that the goal is to perform SVD 1 on $M^T N$ to obtain the singular values for subsequent algebraic operations. In order to achieve such a goal, we need to be able to simulate $\exp(-iM^T Nt)$, followed up with a phase estimation procedure (similar to the HHL algorithm [5]) to 'extract' the singular values and perform our desired

calculation from these values. However, as $M^T N$ is not necessarily symmetric, and hence might not be diagonalizable. We thus aim to simulate $\exp(-i(M^T N)^T M^T N t)$ instead. The reason is that, while $M^T N$ is not symmetric, $(M^T N)^T M^T N$ is symmetric, and hence being diagonalizable, with the eigenvalues of $(M^T N)^T M^T N$ are the square of the singular values of $M^T N$. Furthermore, as we mentioned in a previous section, the singular values of $M^T N$ are guaranteed to be positive, and hence, we will not suffer from the sign problem if we take the square root of eigenvalues of $(M^T N)^T M^T N$.

The following section is devoted to the simulating of $\exp(-i(M^T N)^T M^T N t)$.

A. Encoding Matrix

We first construct a block encoding unitary of $(M^T N)^T M^T N$. For simplicity, let $K \equiv (M^T N)^T M^T N$. It is worth noting that the entries K_{ij}'s are exactly the elements in the i-th column of $M^T N$ multiplied with those in the j-th column of $M^T N$. Since M and N are not necessarily symmetric, we perform the same trick suggested in [5], that is, instead of working with M and N, we work with their Hermitian embedding matrix $\tilde{M}$ and $\tilde{N}$ (both of dimension $(n+k) \times (n+k)$) via

$$\tilde{M} = \begin{pmatrix} 0 & M \\ M^T & 0 \end{pmatrix}, \quad \tilde{N} = \begin{pmatrix} 0 & N \\ N^T & 0 \end{pmatrix}. \tag{7}$$

It is straightforward to see that

$$\tilde{M}\tilde{N} = \begin{pmatrix} MN^T & 0 \\ 0 & M^T N \end{pmatrix}.$$

Without loss of generality, we assume that both $(n+k)$ and k to be a power of 2 (since we can always pad them with extra zero entries), and let $(n+k) = 2^p k$, which is due to our assumption that both $(n+k)$ and k are both some powers of 2. Before proceeding further, we note the following: if $|i\rangle$ is i-th computational basis state, then $A|i\rangle$ is the i-th column of the matrix A. Note that lemma 2 allows us to 'apply' A efficiently, resulting in $A|i\rangle$ entangled with $|0^m\rangle$. Therefore, if $|j\rangle$ is some j-th basis state in the Hilbert space of dimension k, then $\tilde{M}\tilde{N}|2^p\rangle|j\rangle = |2^p\rangle M^T N|j\rangle$. Therefore, by virtue of lemma 2, we have an efficient procedure to implement the following unitary:

$$U_K |\mathbf{0}\rangle |j\rangle = |0^m\rangle |0^m\rangle |2^p\rangle (M^T/s)N |j\rangle + |Garbage\rangle, \tag{8}$$

where $s = s_M s_N$, i.e., the product of the sparsity of M times the sparsity of N. One may wonder why there are two registers $|0^m\rangle$'s. It comes from the fact that we consecutively use lemma 2 to apply $\tilde{M}$ and $\tilde{N}$. In order to construct the unitary encoding of K, we need to slightly modify U_K to obtain different unitaries $U_{K,1}$ and $U_{K,2}$ in the following ways.

- Beginning with $U_k |\mathbf{0}\rangle |j\rangle$, we append another ancilla initialized in $|00\rangle$:

$$|0^m\rangle |0^m\rangle |2^p\rangle (M^T N/s) |j\rangle |00\rangle + |Garbage\rangle |00\rangle.$$

Use X gate on the last bit to flip $|0\rangle$ to $|1\rangle$, and apply $C^{2m+1} X$ conditioned on the first two registers being $|0^m\rangle |0^m\rangle$ and last bit being $|1\rangle$ (as the control) to flip the next-to-last qubit (as the target) to obtain:

$$|0^m\rangle |0^m\rangle |2^p\rangle (M^T N/s) |j\rangle |01\rangle + |Garbage\rangle |11\rangle.$$

We denote the above process as $U_{K,1}$.

- With the state above, we apply a CNOT using the next-to-last bit as the control bit to flip the last bit, and we obtain:

$$|0^m\rangle |0^m\rangle |2^p\rangle (M^T N |j\rangle /s) |01\rangle + |Garbage\rangle |10\rangle.$$

We denote the combination of $U_{K,1}$ plus the above extra step as $U_{K,2}$. It is easy to see the following:

$$\langle \mathbf{0}| \langle i| U_{K,2}^\dagger U_{K,1} |\langle 0|\rangle |j\rangle = \frac{1}{s^2} \langle i| (M^T N)^T M^T N |j\rangle = K_{ij}/s^2. \tag{9}$$

The above property matches with the definition of block encoding 1. Therefore, we have that $U_{K,2}^\dagger U_{K,1}$ is the unitary block encoding of $K \equiv (M^T N)^T M^T N$ divided by s^2.

B. Computing Distance d_{M_A,M_B}

As we have already succeeded in producing the unitary block encoding of K, it is quite straightforward to apply the tools in Lemma 1 to simulate $\exp(-iKt)$. We have the following result.

Lemma 3 *The evolution* $\exp(-i(M^T N)^T M^T N t)$ *can be simulated to accuracy* ϵ_H *in time*

$$\mathcal{O}\Big(s^2 log(n+k)(t+log(\frac{1}{\epsilon_H}))\Big), \tag{10}$$

where $s = s_M s_N$.

The ability to simulate $\exp(-i(M^T N)^T M^T N t)$, combined with the QPE subroutine [30, 31] yields the ability to extract the eigenvalues of $(M^T N)^T M^T N$. This method has been used in the famous HHL algorithm [5] to invert a given matrix, hence yielding an efficient quantum algorithm for solving linear systems. In order to compute the Grassmann distance, we need to make some modifications, as we will work with mixed states instead of pure states as in [5]. However, the analysis regarding error provided in [5] still holds in the general case, and we will soon exploit it to prove the efficiency and error bound of our distance calculation procedure.

The distance calculation procedure begins with the following mixed state:

$$\rho = \frac{1}{k} \sum_{i=1}^{k} |i\rangle \langle i|, \tag{11}$$

which can be easily prepared by applying $H^{\log(k)} \otimes I^{\log(k)}$ to $|0\rangle^{\log(k)} \otimes |0\rangle^{\log(k)}$, followed by CNOT layers composed of $\log(k)$ CNOT gates, then tracing out either register. Since ρ is diagonal and proportional to the identity matrix in the computational basis state, it is also diagonal in the basis containing eigenvectors of $(M^T N)^T M^T N$. We recall that since $(M^T N)^T M^T N$ is symmetric, its eigenvectors are mutually orthogonal. Let $\{|u_i\rangle, \lambda_i\}_{i=1}^{k}$ denotes these eigenvectors and eigenvalues. We use the following important relations and formulas:

- $\lambda_i = \sigma_i^2$ for all i, where σ_i is the singular value of $M^T N$.

- $0 \leq \sigma_i \leq 1$ for all i, i.e., σ_i is non-negative for all i.

- Grassmann distance:

$$d_{M_A,M_B} = \sqrt{\sum_{i=1}^{k} \theta_i^2}, \tag{12}$$

where $\cos(\theta_i) = \sigma_i$. We remind the readers that since $0 \leq \sigma_i \leq 1$, $\theta_i = \arccos(\sigma_i)$ can be chosen to lie within the range $(0, \pi/2)$ for all i. Therefore, the Grassmann distance can be rewritten as:

$$d_{M_A,M_B} = \sqrt{\sum_{i=1}^{k}(\arccos(\sigma_i))^2} = \sqrt{\sum_{i=1}^{k}(\arccos(\sqrt{\lambda_i}))^2}. \tag{13}$$

In order to compute the above distance, we first run the QPE, with varied time unitary $\exp(-i(M^T N)^T M^T N t/C^2)$ and ρ as the input, in a similar fashion to the first part of the HHL algorithm [5]. More precisely, we make use of lemma 3 to apply the controlled evolution (as in [5])

$$\sum_{\tau} |\tau\rangle \langle\tau| \otimes \exp(-i(M^T N)^T M^T N \tau t_0/T), \tag{14}$$

for varying τ (the register that holds $|\tau\rangle$ is called the phase register) and a fixed T to the following state as part of the QPE subroutine:

$$\frac{1}{T} \sum_{\tau,\tau'=0}^{T-1} |\tau\rangle \langle\tau'| \otimes \rho, \tag{15}$$

followed by an inverse quantum Fourier transform on the phase register. Ideally, if the QPE is exact, we would obtain the following state:

$$\frac{1}{k}\sum_{i=1}^{k}|\lambda_i\rangle\langle\lambda_i|\otimes|u_i\rangle\langle u_i|. \tag{16}$$

Now we append another ancilla initialized as $|0\rangle$ (technically, it should be written $|0\rangle\langle0|$ as we are dealing with mixed states; however, it does not possess any systematic issue), performing the following rotation controlled by the phase register $\{|\lambda_i\rangle\}$

$$|0\rangle\rightarrow\Big(\frac{\arccos(\sqrt{\lambda_i})}{(\pi/2)}|0\rangle+\sqrt{1-(\frac{\arccos(\sqrt{\lambda_i})}{(\pi/2)})^2}\,|1\rangle\Big), \tag{17}$$

we obtain:

$$\frac{1}{k}\sum_{i=1}^{k}|\lambda_i\rangle\langle\lambda_i|\otimes|u_i\rangle\langle u_i|\otimes\Big(\frac{\arccos(\sqrt{\lambda_i})}{(\pi/2)}|0\rangle+\sqrt{1-(\frac{\arccos(\sqrt{\lambda_i})}{(\pi/2)})^2}\,|1\rangle\,\Big)\Big(\frac{\arccos(\sqrt{\lambda_i})}{(\pi/2)}\langle0|+\sqrt{1-(\frac{\arccos(\sqrt{\lambda_i})}{(\pi/2)})^2}\,\langle1|\,\Big). \tag{18}$$

The above state seems somewhat complicated. However, we only need to pay attention to the part entangled with the state $|0\rangle\langle0|$ on the ancilla. If we make a measurement on the ancilla, the probability of measuring $|0\rangle\langle0|$ is:

$$p_0=\frac{4}{\pi^2k}\sum_{i=1}^{k}(\arccos(\sqrt{\lambda_i}))^2=\frac{4}{\pi^2k}d^2_{M_A,M_B}. \tag{19}$$

Hence, once we can estimate p_0, for example, by repeating the measurement and counting the frequency of seeing 0, then we can estimate d_{M_A,M_B}. In order to estimate p_0 to accuracy δ^2, and d_{M_A,M_B} to accuracy $\mathcal{O}(\delta)$, we need to repeat the measurement $\mathcal{O}(1/\delta^2)$ times.

C. Error Analysis

The previous section assumes an ideal case, as we have emphasized, that when the QPE is exact. In reality, there is an error due to finite-bit precision in the QPE, which means that we only obtain the approximated phase, $\tilde{\lambda}_i$, instead of λ_i. Fortunately, this critical issue has been analytically worked out and dealt with in [5]. We strongly refer the readers to [5] for a complete treatment, as we will not repeat the calculation here. Rather, we directly apply their analysis to our context. Denote κ as the conditional number of M^TN (which means that κ^2 is the conditional number of $(M^TN)^TM^TN$), if we have

$$t_0=\mathcal{O}(\kappa^2/\epsilon_p),$$

where t_0 is a fixed term appearing in the conditional evolution, i.e., in Eqn. 14. Then the analysis from [5] applied here yields the following bound on the error:

$$\sqrt{\frac{4}{\pi^2k}\sum_{i=1}^{k}\Big(\arccos(\sqrt{\tilde{\lambda}_i})-\arccos(\sqrt{\lambda_i})\Big)^2}\leq\epsilon_p. \tag{20}$$

The finite-bit precision in the QPE further affects the actual measurement outcome. We summarize our result in the following lemma.

Lemma 4 *In non-ideal case, the probability of measuring $|0\rangle\langle0|$ now becomes:*

$$\tilde{p}_0=\frac{4}{\pi^2k}\sum_{i=1}^{k}(\arccos(\sqrt{\tilde{\lambda}_i}))^2, \tag{21}$$

and

$$\Big|\tilde{p}_0-p_0\Big|\leq\epsilon_p. \tag{22}$$

Proof of the above lemma is given in Appendix A. It essentially means that in the non-ideal case, we can only estimate an approximated value of p_0, or equivalently, d_{M_A,M_B}, as $d_{M_A,M_B} = \frac{1}{2}\pi\sqrt{k}\sqrt{p_0}$.

As we have outlined a quantum algorithm for computing linear subspace distance, we have seen many sources of error throughout the whole procedure. For the purpose of summary, we revise the main steps of our algorithm with the corresponding error contribution from such steps before establishing the final running time with respect to overall error tolerance, for which we eventually set to ϵ:

- **State Preparation:** Error ϵ_S induced from matrix application step 2. In principle, ϵ_S would contributes to the simulation of error of $\exp(-i(M^T N)^T M^T N t)$. However, the running time of the preparation step scales as $\mathcal{O}(\log(1/\epsilon_S))$, which is very efficient. Therefore, we can neglect the error contribution from this step, as we can make it very small at a modest cost.

- **Simulating Evolution:** Error ϵ_H induced directly from improper simulation of $\exp(-i(M^T N)^T M^T N t)$, as a result of density matrix exponentiation technique [32]. As we stated in lemma 3, the running time scales $\mathcal{O}(log(1/\epsilon_H))$, which is logarithmic and hence, is efficient. We remind that our algorithm shares a similar routine as the HHL algorithm [5], but in their work, the error induced from simulating the evolution of a given matrix, say, $\exp(-iHt)$ is negligible.

- **Performing Quantum Phase Estimation:** The error is resulted from the improper phase estimation ϵ_P. This is probably the most complicated factor in our algorithm. Inaccurate phase estimation results in an inaccurate distance formula, as we point out in equation 21.

- **Estimating Distance d_{M_A,M_B} From Measurement Outcome:** Its error δ comes from the estimation of $\tilde{p_0}$, which gives a further running time $\mathcal{O}(1/\delta^2)$ as a result of the Chernoff bound. We remark that the quantum amplitude estimation [31] can improve the cost to $\mathcal{O}(1/\delta)$. This step can be done with $\mathcal{O}(1)$ time by preparing $\mathcal{O}(1/\delta^2)$ identical quantum circuits and running them in parallel before averaging the result statistically.

All in all, the most dominant error comes from the phase estimation step, similar to [5]. For simplicity, we can set the desired error tolerance to be ϵ. We establish our first main result.

Theorem 1 (Estimation of Grassmann Distance) *Given access to matrix M and $N \in R^{n \times k}$ as defined in section II. Let κ denote the conditional number of $M^T N$. Then the Grassmann distance between M_A and M_B, which is spanned by column vectors of M and N, respectively, can be estimated to additive accuracy ϵ in time*

$$\mathcal{O}\left(\kappa^2 s^2 \log(n+k) \cdot \frac{1}{\epsilon^2}\right).$$

This is a general statement about the running time of the algorithm that we have outlined. Now, we will discuss some aspects of the algorithm, and particularly, we will show that there are specific scenarios that achieve a better running time.

D. Comments

Conditional Number: The dependence on conditional number κ is worth examining further. The concern is, when will κ be small? Since M and N are formed by orthogonal vectors that could be drawn from random, it is generally hard to predict how κ would behave. However, in the above paragraph, we have described a geometric picture about the entries of $M^T N$, which is essentially the inner product of corresponding columns of M and N. We now mention the following interesting result regarding the conditional number of a matrix.

Theorem 2 (Theorem 1 in [33] (Lower Bound on Singular Value)) *Let A be $n \times n$ matrix and assumed to be diagonally dominant by rows and set $\alpha = \min_k(|a_{kk}| - \sum_{j \neq k} |a_{jk}|)$. Then $\sigma_{min} > \alpha$.*

The above result can equivalently work if A is diagonally dominant by columns. Recall that the diagonal entries $(M^T N)_{ii}$ are the inner product of i-th column of M and N. What does it mean for $M^T N$ to be diagonally row/column dominant? If for every i, we have the condition that m_i is 'very close' to n_i, and quickly becomes almost orthogonal to the remaining vector from $\{n\}$ as the index runs away from i, then we can guarantee that the matrix $M^T N$ is diagonally row dominant. By virtue of the above theorem, we have that the minimum singular value of $M^T N$ is lower bounded by a constant (independent of dimension), which directly implies that its conditional number κ is upper

bounded by a constant.

Advantage Over Classical Algorithm: We have mentioned in Section II that the seemingly best classical algorithm for computing Grassmann distance takes time

$$\mathcal{O}(n^2 k + k^3). \tag{23}$$

Compared with the quantum running time (Theorem 1), we see that if $\kappa, s \ll n, k$, for example, are of order $\approx \mathcal{O}(\log(n,k))$, then there is an exponential speedup with respect to both n and k. In the previous paragraph, we have mentioned a setting where κ can be small. Recall that $s = s_M s_N$, and, therefore, our quantum algorithm performs the best when both M and N are sparse. In the dense regime, i.e., $s \approx \max(n,k)^2$, and this means that with respect to (n,k), the quantum running time could be as much as $\mathcal{O}(\max(n,k)^4)$, which is much smaller than the classical algorithm, especially when $\max(n,k) = n$. The above running time holds for all cases. Because even when both M and N are sparse, their product is not guaranteed to be sparse. Therefore, one may still need to perform SVD on a dense matrix.

V. QUANTUM ALGORITHM FOR COMPUTING ELLIPSOID DISTANCE

Now we turn our attention to the second problem: computing the distance between two ellipsoids $\mathcal{E}_M$ and $\mathcal{E}_N$, each defined by a real symmetric positive definite matrix M and $N \in R^{n \times n}$, respectively. Without loss of generality, we assume eigenvalues of M and N to be bounded in the known range $(1/\kappa, 1)$, which is always achievable by rescaling. To recall, such distance is calculated as follows:

$$\delta_{M,N} = \sqrt{\sum_{i=1}^{n} \log^2(\lambda_i(M^{-1}N))}. \tag{24}$$

While at first, the symmetric property of both M (and hence of M^{-1}) and N may seem useful; however, this does not make calculating the above distance trivial. The reason is that $M^{-1}N$ is not necessarily symmetric, which can be seen by simply performing the transpose:

$$(M^{-1}N)^T = N^T (M^{-1})^T = NM^{-1},$$

which is different from $M^{-1}N$ in general.

A. Encoding Matrix

For convenience, we first set $P = M^{-1}N$. In order to resolve the non-symmetric issue, we use the same trick as we did in last section, i.e., we would try to simulate $\exp(-iP^T P t)$, then doing algebraic operations on its eigenvalues. This problem is somewhat more challenging than the Grassmann distance, as there is a requirement for matrix inversion. This is exactly the utility of the celebrated quantum linear solver algorithm [5, 6]. We remark that, in our case, M can be dense. The inversion of a dense matrix has been done in [24], with a particular quantum data structure. That kind of quantum data structure is not assumed in our case, as we only work with the familiar blackbox model. Therefore, we would use the result of [6] to achieve the matrix inversion since this method achieves better scaling time on error tolerance.

Lemma 5 (Matrix Inversion [6]) *Given access to some Hermitian matrix M of size $n \times n$, an initial state $b \equiv |b\rangle$. There is a unitary that performs the following map:*

$$U_M |0\rangle |b\rangle = |0\rangle |M^{-1}b| |M^{-1}b\rangle + |1\rangle |Garbage\rangle. \tag{25}$$

The running time of U_M is $\mathcal{O}\left(s_M \log(n)\kappa_M poly(\log 1/\epsilon))\right)$ where s_M is the sparsity of matrix M, ϵ is some tolerance error (the state $|M^{-1}b| |M^{-1}b\rangle$ is just an approximation of the state).

With the above result and combined with lemma 2, we are able to create the state that corresponds to columns of P. More specifically, we use Lemma 2 to achieve the following:

$$U_N |0^m\rangle |i\rangle = |0^m\rangle (N/s_N) |i\rangle + |Garbage\rangle, \tag{26}$$

where s_N is the sparsity of N. For a reason that would be clear later, we append another ancilla $|0\rangle$ and work in a larger Hilbert space. We would have instead:

$$\mathbb{I} \otimes U_N |0\rangle |0^m\rangle |i\rangle = |0\rangle |0^m\rangle (N/s_N) |i\rangle + |0\rangle |Garbage\rangle. \tag{27}$$

Next, we use $|0^m\rangle$ as the control system to flip the first qubit, i.e., we transform the above state to:

$$|1\rangle |0^m\rangle (N/s_N) |i\rangle + |0\rangle |Garbage\rangle.$$

We then use matrix inversion from Lemma 5, controlled by the first $|1\rangle$ to invert M on the state $(N/s_N) |i\rangle$. To be more precise, we need $|0\rangle (N/s_N) |i\rangle$ in order for U_M to be effective. The extra $|0\rangle$ can be borrowed from $|0\rangle^m$, i.e, we have $|0\rangle^m (N/s_N) |i\rangle \equiv |0\rangle^{m-1} |0\rangle (N/s_N) |i\rangle$. We further note that this time the unitary U_M in 5 is controlled by a qubit being $|1\rangle$ (the first register in the above state). We obtain the following unitary denoted as U:

$$U |0\rangle |0^m\rangle |i\rangle = |1\rangle |0\rangle^{m-1} \left(|0\rangle M^{-1}(N/s_N) |i\rangle + |1\rangle |Garbage_1\rangle \right) + |0\rangle |Garbage\rangle. \tag{28}$$

Since the role of all garbage states is the same, as they do not contribute to the final result, we simplify the above representation as:

$$|1\rangle |0\rangle^m M^{-1} N/s_N |i\rangle + |Garbage\rangle. \tag{29}$$

Now, we again use the m-qubit register as control and, conditioned on they being $|0^m\rangle$, flip the first qubit back to $|0\rangle$. The final state becomes

$$|0\rangle |0^m\rangle M^{-1}(N/s_N) |i\rangle + |Garbage\rangle, \tag{30}$$

which has a similar form as 8. Therefore, we can use the same procedure, with extra two ancilla qubits, as we outlined in section IV A to block encode matrix $C^2 P^T P/s_N^2$ (recall that $M^{-1}N |i\rangle$ is the i-th column of $M^{-1}N$ and we have set $P \equiv M^{-1}N$). In a clear manner, the application of Lemma 1 yields the following:

Lemma 6 *The simulation of* $\exp(-i(M^{-1}N)^T M^{-1}Nt)$ *can be achieved up to accuracy ϵ in*

$$\mathcal{O}\left(s_M s_N^2 \, poly \log(n, \frac{1}{\epsilon}) \cdot \kappa_M t \right).$$

B. Computing Ellipsoid Distance

Once we can simulate $\exp(-iP^T Pt)$, we then run the QPE with completely mixed state $\rho = (1/n) \sum_{i=1}^n |i\rangle \langle i|$ as the input. After the QPE, we would obtain a state similar to Eqn. 16. We then append an ancilla $|0\rangle$ and rotate as the following (again, we are ignoring the mixed state formalism):

$$|0\rangle \rightarrow \log(\tilde{\lambda}_i) |0\rangle + \sqrt{1 - \log(\tilde{\lambda}_i)^2} |1\rangle, \tag{31}$$

where $\tilde{\lambda}_i$ is the approximated i-th eigenvalue of $P \equiv M^{-1}N$. We then measure the ancilla, with the probability of measuring $|0\rangle$ could be shown to be:

$$p = \sum_{i=1}^n \log^2(\tilde{\lambda}_i)/n = \tilde{\delta}_{M,N}^2/n, \tag{32}$$

where $\tilde{\delta}_{M,N}^2$ denotes the approximated value of the real ellipsoids distance. Following the same analysis as in previous section (which is based on analysis in the HHL algorithm), if we choose $t = \mathcal{O}(\kappa_P/\epsilon)$ (where κ_P is the conditional number of P) in the simulation of $\exp(-iP^T Pt)$, then we can guarantee an overall error ϵ, i.e, $|\tilde{\delta}_{M,N} - \delta_{M,N}| \leq \epsilon$. The value of p itself can be estimated to accuracy ϵ by repeating the measurement $\mathcal{O}(1/\epsilon^2)$ times, which can be improved to $\mathcal{O}(1/\epsilon)$ by employing quantum amplitude estimation [31]. We now establish our results regarding the estimation of the ellipsoid distance.

Theorem 3 (Estimation of Ellipsoids Distance) *Let $\mathcal{E}_M$ and $\mathcal{E}_N$ be ellipsoids defined by two real symmetric positive definite matrix M and N. Given local access to entries of M and N, the distance between $\mathcal{E}_M$ and $\mathcal{E}_N$, denoted as $\delta_{M,N}$, can be estimated in time:*

$$\mathcal{O}\Big(s_M s_N^2 \log(n) \frac{\kappa_P \kappa_M}{\epsilon^2}\Big),$$

where κ_P is the conditional number of $M^{-1}N$.

One may wonder that there seems to be a missing factor of order $\mathcal{O}(polylog(1/\epsilon))$, as it appears in the running time of Lemma 6. The main reason is that we simply absorb that scaling into the $\mathcal{O}(1/\epsilon^2)$, which is a result of Hadamard estimation plus an HHL-like procedure.

   Advantage Over Classical Algorithm: As we have mentioned from Sec. II, the best classical algorithm for computing ellipsoids distance has running time $\mathcal{O}(n^3)$, with the running time dominated by performing classical SVD. If both M and N are sparse, with low conditional numbers (of order $\log(n)$), then there is exponential speedup. Even when matrix M is not sparse, i.e., $s_M \in \mathcal{O}(n)$, but N is sparse and $M^{-1}N$ has low conditional number, there is polynomial speedup.

   In the Grassmann distance case, we have pointed out the specific scenario where the conditional number of the corresponding matrix could be low; however, it is quite difficult to find such a case in the ellipsoid distance case. The reason is apparently due to the inversion M^{-1}. If M is orthogonal, i.e., $M^{-1} = M^T$, then it becomes similar to the Grassmann distance case, where we can have a specific scenario with a low conditional number.

VI.   EXTENSION TO DIFFERENT KIND OF DISTANCES

   The above two main quantum algorithms were devoted to estimating the so-called Grassmann distance and ellipsoid distance. Here we aim to extend the application of our quantum algorithm, specifically the Grassmannian case, by discussing some other related distances that could also be practically useful [23]. We note that we would adopt the same notations as in section IV.

   The first one is called *Asimov distance*, which is defined as following:

$$d_{M,N}^A = \theta_k, \tag{33}$$

where $\cos(\theta_k) = \sigma_k$ is the smallest singular value of $M^T N$. Therefore, we first need to find the value of σ_k. Finding the maximum and minimum eigenvalues of a given Hermitian matrix has been done in [34], based on the (classical) power method. In our case, we are given the ability to simulate $\exp(-i(M^T N)^T M^T N t)$. Still, the method of [34] can be adapted in a straightforward manner. It will yield us the approximated minimum eigenvalue of $(M^T N)^T M^T N$, which in turn gives an estimation of σ_k^2. As was analyzed in [34], the random initialization step is quite critical in the performance of the algorithm, as it might diminish the exponential speedup. In general, only quadratic speedup is obtained.

   The next one, very close to the Asimov distance, is called *projection distance*, which is defined simply as:

$$d_{M,N}^P = \sin(\theta_k), \tag{34}$$

where the angle θ_k is defined in the same way as in the Asimov distance case. It is easy to see that $(d_{M,N}^P)^2 = 1 - \cos(\theta_k)^2 = 1 - \sigma_k^2$. Therefore, estimating σ_k^2 is sufficient. Such estimation of σ_k was just described above by using the method in [34]. Thus, its efficiency is the same as that of the Asimov distance.

   Now we discuss the *Chordal distance*, which is defined as follows:

$$d_{M,N}^C = \Big(\sum_{i=1}^{k} \sin(\theta_i)^2 \Big)^{1/2}. \tag{35}$$

We note the following:

$$(d_{M,N}^C)^2 = \sum_{i=1}^{k} \sin(\theta_i)^2 \tag{36}$$

$$= k - \sum_{i=1}^{k} \cos(\theta_i)^2 \tag{37}$$

$$= k - \sum_{i=1}^{k} \sigma_i^2. \tag{38}$$

Therefore, it suffices to estimate $\sum_{i=1}^{k} \sigma_i^2$. This can be done in a very similar manner to Grassmann distance estimation, except that in the rotation step 17, we do not have to do any further arithmetic operation, which makes it even simpler. Therefore, the estimation of the Chordal distance can be done with the same running time as that of the Grassmann distance; see Theorem 1.

VII. ESTIMATING GRASSMANN DISTANCE AND ELLIPSOID DISTANCE IN THE MEMORY MODEL

In the above sections, we work in the standard blackbox model that assumes coherent access to entries of corresponding matrices. Here, we explore how the so-called memory model can potentially enhance the estimation of Grassmann and ellipsoid distances. Particularly, we shall see that the quantum running time in the memory model is sparsity-independent. This is within our expectation, as in [24], the authors also used such a model to construct a quantum linear solver that has running time independent of the sparsity of the given matrix.

To begin with, the memory model was proposed in [35] as a novel quantum architecture that allows efficient load out of classical data. In the standard blackbox model, data entries are accessible individually, whereas, in the memory model, data is usually loaded column/row-wise. In [35], the authors showed how this model can give rise to an efficient quantum algorithm for the recommendation system. In particular, this model, combined with the famous quantum phase estimation algorithm, yields a sparsity-independent quantum linear solver [24]. While the problem of recommendation system was "de-quantized" in the seminal work of Tang [36] (where the author showed that under appropriate assumption regarding input access, there exists a classical algorithm that could solve the recommendation system problem with at most a polynomial slowdown compared to the quantum counterpart), the advantage of a dense quantum linear solver seems to hold due to BQP-completeness of matrix inversion [5]. Throughout this work, we have seen that the blackbox model could yield an efficient quantum algorithm for estimating different kinds of geometric distances. Thereby, it is very interesting to expand the potential of the memory model in solving various computational tasks other than dense linear systems [24].

Before diving into the algorithms, we first recall the features of the memory model.

Lemma 7 (Data Structure [24, 35]) *Given a matrix $M \in \mathbb{R}^{n \times m}$. Then there exists a quantum data structure that allows the following coherent mapping in $\mathcal{O}(poly(\log(mn)))$ time:*

$$U_M \ket{\mathbf{i}} \ket{0} \to \frac{1}{||A_i||} \sum_{j=1}^{n} A_{ij} \ket{j} \ket{i},$$

$$U_N \ket{0} \ket{j} \to \frac{1}{||A||_F} \sum_{i=1}^{m} ||A_i|| \ket{i} \ket{j},$$

where i, j refers to the row and column index of A.

Similar to our previously described algorithm in the blackbox model, we will employ the above mapping to construct the block encoding of corresponding matrices, followed by the QPE and measurement to extract the desired distances.

A. Grassmann Distance

Recall that in the Grassmann distance problem, the algorithm outlined in section IV relies on the simulation of $\exp(-i(M^T N)^T M^T N t)$. The question now is how to perform block-encoding of $M^T N$ given the memory model structure in Lemma 7. We first remind that the matrix M^T is of size $k \times n$, and the matrix N is of size $n \times k$, where n is the dimension of given data and k is the number of data points. WOLG, for simplicity, we embed M and N into a bigger matrix of size $\max(k,n) \times \max(k,n)$, with those additional entries set to 0. With such simplification, we would work with a square matrix instead, which is more convenient.

Now, we attempt to do block-encoding of $M^T N$ given the memory model structure. We note that, as mentioned in [24], the memory model naturally provides the block-encoding of $M/|M|_F$ and $N/|N|_F$ (we shall derive this in the appendix), where $|.|_F$ refers to the Frobenius norm of these matrices. For convenience, we denote those block-encoding unitaris as U_M and U_N, respectively.

We first remind from definition 1 that if a matrix A is encoded in some unitary U, then it can be written as:

$$U = |\mathbf{0}\rangle \langle \mathbf{0}| \otimes A + \cdots \tag{39}$$

It is worthy to observe that, the above unitary U acts on the state $|\mathbf{0}\rangle |\phi\rangle$, where $|\phi\rangle$ has the same dimension as A, as following:

$$U |\mathbf{0}\rangle |\phi\rangle = |\mathbf{0}\rangle A |\phi\rangle + \sum_{j \neq \mathbf{0}} |j\rangle |\phi_j\rangle, \tag{40}$$

where $|\phi_j\rangle$ refers to some redundant state. For some reason that would become clear later, we add an ancilla $|0\rangle$ and produce the following:

$$\mathbb{I} \otimes U |0\rangle |\mathbf{0}\rangle |\phi\rangle = |0\rangle |\mathbf{0}\rangle A |\phi\rangle + |0\rangle \sum_{j \neq \mathbf{0}} |j\rangle |\phi_j\rangle. \tag{41}$$

With the above observation, let $|i\rangle, |k\rangle$ be some arbitrary computational basis states (with their dimensions corresponding to those of the matrices M, N, respectively), we have the following:

$$\mathbb{I} \otimes U_M |0\rangle |\mathbf{0}\rangle |i\rangle = |0\rangle |\mathbf{0}\rangle (M/|M|_F) |i\rangle + |0\rangle \sum_{j \neq \mathbf{0}} |j\rangle |\phi_j\rangle \equiv |\Phi_M\rangle, \tag{42}$$

$$\mathbb{I} \otimes U_N |0\rangle |\mathbf{0}\rangle |k\rangle = |0\rangle |\mathbf{0}\rangle (N/|N|_F) |k\rangle + |0\rangle \sum_{j \neq \mathbf{0}} |j\rangle |\phi_j\rangle \equiv |\Phi_N\rangle. \tag{43}$$

- Now for state $|\Phi_N\rangle$, we use X gate to flip the ancilla, e.g., the first qubit to obtain

$$X \otimes \mathbb{I} |\Phi_N\rangle = |1\rangle |\mathbf{0}\rangle (N/|N|_F) |k\rangle + |1\rangle \sum_{j \neq \mathbf{0}} |j\rangle |\phi_j\rangle \equiv |\Phi_N^1\rangle. \tag{44}$$

- Now we use the register $|\mathbf{0}\rangle$ as a control register to flip the first qubit back to $|0\rangle$ (i.e., flip conditioned the register being $\mathbf{0}$), and we obtain the state:

$$|0\rangle |\mathbf{0}\rangle (N/|N|_F) |k\rangle + |1\rangle \sum_{j \neq \mathbf{0}} |j\rangle |\phi_j\rangle \equiv |\Phi_N^2\rangle. \tag{45}$$

Denote the unitary $\mathbb{I} \otimes U_N$ plus the above two additional steps as P_N. As previously pointed out, for arbitrary matrix A, $A |i\rangle$ is the i-th column of A. Furthermore, the entries of matrix $M^T N$ are basically the inner product of columns of N and M. It is then straightforward to observe that:

$$\langle \Phi_M | \Phi_N^2 \rangle = \langle i| \frac{M^\dagger}{|M|_F} \frac{N}{|N|_F} |k\rangle = \frac{(M^T N)_{ik}}{|M|_F |N|_F}, \tag{46}$$

where we use that M, N are real then $M^\dagger = M^T$, which is just the transpose. The above property matches perfectly with the definition of block-encoding 1. Therefore, the unitary $(\mathbb{I} \otimes U_M^\dagger)(P_N)$ is exactly the unitary block-encoding

of $(M^T N)/|M|_F |N|_F$. Note that this unitary has a larger dimension than the initial unitary encoding of M and N.

Given the block-encoding of $M^T N/|M|_F |N|_F$, it is very easy to perform the block encoding of $(M^T N)^T (M^T N)/(|M|_F |N|_F)^2$. Basically, we can apply the same procedure as we just outlined. Therefore, we do not repeat it here. We simply would proceed with the following lemma:

Lemma 8 *In presence of the memory model with given data structure γ of M and N, the simulation of $\exp(-i(M^T N)^T M^T N t)$ can be achieved up to accuracy ϵ in time*

$$\mathcal{O}\Big(polylog((n,k))|N|_F^2 |M|_F^2 (t + \log(1/\epsilon))\Big),$$

where we remark that (n, k) refers to $\max(n, k)$. Using such ability, one can apply essentially the same algorithm outlined in IV B to estimate the Grassmann distance. Therefore, we have the following straightforward statement:

Theorem 4 *Given access to matrices M and N in the memory model, the Grassmann distance between M_A and M_B can be estimated to additive accuracy δ in time*

$$\mathcal{O}\Big(\kappa^2 |M|_F^2 |N|_F^2 polylog((n,k)) \cdot \frac{1}{\delta^2}\Big).$$

Now, we make the following comparison. Since both M and N are of size $n \times k$ and are assumed to have columns of unit norm, their Frobenius norm is $\sqrt{k}$. Therefore, the actual quantum running time is $\mathcal{O}\Big(\kappa^2 k^2 polylog((n,k)) \cdot \frac{1}{\delta^2}\Big)$. If the conditional number κ does not grow anything faster than $polylog((n,k))$, this running time is considerably much shorter than the classical running time $\mathcal{O}(n^2 k + k^3)$, as well as the quantum algorithm in the blackbox model (see Sec IV B) in the dense setting, which could be $\mathcal{O}(k^4)$.

B. Ellipsoid Distance

Now, we discuss how ellipsoid distance could be estimated in the memory model. Recall that in the ellipsoids distance problem, we are given two symmetric positive definite matrices M and N, and we need to estimate the following quantity:

$$\delta_{M,N} = \sqrt{\sum_{i=1}^{n} \log^2(\lambda_i(M^{-1} N))}, \tag{47}$$

For completeness, we first recall a few key operations as well as important observations.

The first one is, as we also pointed out in the last section, the memory model naturally produces the block encoding unitaries of $M/|M|_F$ and $N/|N|_F$, respectively. We denote them here as U_M and U_N for simplicity.

The second one is the operation of U_M (respectively U_N) on the given arbitrary state $|\mathbf{0}\rangle |\phi\rangle$:

$$U |\mathbf{0}\rangle |\phi\rangle = |\mathbf{0}\rangle A |\phi\rangle + \sum_{j \neq \mathbf{0}} |j\rangle |\phi_j\rangle. \tag{48}$$

The last one is the matrix inversion quantum algorithm that was proposed in [24].

Lemma 9 *In the presence of the memory model (see γ), given some initial state $|b\rangle$, the following unitary could be implemented:*

$$U_{invert}^M |0\rangle |b\rangle = |0\rangle CM^{-1} |b\rangle + |1\rangle |Garbage\rangle. \tag{49}$$

The running time of U_{invert}^M is

$$\mathcal{O}\Big(\kappa_M |M|_F \cdot polylog(n) \frac{1}{\epsilon}\Big),$$

where κ_M is the conditional number of M; $|M|_F$ is the Frobenius norm of M and ϵ is the error tolerance; and C is the factor that is required for the normalization purpose, e.g., $C \leq 1/\kappa$.

Now, we are ready to describe our quantum algorithm in the memory model. It turns out that the quantum algorithm in this case (memory model) is essentially similar to that of blackbox model (see Section V). As we shall see, the only difference is the matrix inversion step, where the memory model supports a faster subroutine.

We begin with some computational basis state $|i\rangle$ (which shares the same dimension as M, N), we use U_N to act and obtain:

$$\mathbb{I} \otimes U_N |0\rangle |\mathbf{0}\rangle |j\rangle = |0\rangle |\mathbf{0}\rangle (N/|N|_F) |i\rangle + |0\rangle \sum_{j \neq \mathbf{0}} |j\rangle |\phi_j\rangle. \tag{50}$$

We make an observation that this state is very similar to Eqn. 26. Following the same procedure as in V (basically the whole paragraph after equation 26), we use $|\mathbf{0}\rangle$ to flip the first qubit $|0\rangle$ to $|1\rangle$, then use $|1\rangle$ as a controlling qubit to apply Lemma 9. We yield the following state:

$$|0\rangle |0^m\rangle M^{-1} (N/|N|_F) |i\rangle + |Garbage\rangle. \tag{51}$$

Therefore, we have the following:

Lemma 10 *In the presence of memory model, the simulation of* $\exp(-iP^T P t)$ *can be achieved up to accuracy* ϵ *in*

$$\mathcal{O}\Big(\kappa_M |M|_F \cdot polylog(n) \frac{1}{\epsilon} t |N|_F^2\Big).$$

With this, following the same procedure outlined V, the ellipsoid distance can be estimated.

Theorem 5 (Ellipsoid Distance in Memory Model) *In the presence of the memory model, the distance between two ellipsoids defined by two real symmetric positive definite matrices* M *and* N *can be estimated up to an additive accuracy* ϵ *in time:*

$$\mathcal{O}\Big(|M|_F \cdot polylog(n) |N|_F^2 \frac{\kappa_P \kappa_M}{\epsilon^3}\Big),$$

where $P \equiv M^{-1} N$.

VIII. CONCLUSION

Motivated by fast-pace development of quantum algorithm, as well as the prospect of quantum advantage, we have outlined two quantum algorithms for estimating Grassmann distance and ellipsoid distance between two subspaces formed by corresponding data elements. We have specifically constructed quantum algorithm in both data models, which is the standard blackbox model and the newly developed memory model. Our algorithm is built upon density matrix exponentiation [12], fast quantum matrix application [34] and quantum linear solving algorithm [5, 6]. Under the corresponding assumption regarding the input access, as well as appropriate conditions regarding sparsity and conditional number, our quantum algorithms yield significant speedup compared to classical algorithms that could solve the same problems, e.g, estimating Grassmann distance and ellipsoids distance. The novelty of our approach, or more specifically, the use of quantum phase estimation, lies in the way that we perform algebraic operations directly on the phase registers, and essentially combine them to estimate corresponding distances. As the effort for expanding the applicability of quantum computer still goes on, we strongly believe that the techniques outlined here could find more impactful benefits in devising novel quantum algorithm to solve challenging computational problems. As we mentioned, our work have added to a few existing works, e.g, topological data analysis [15], homology detection [21], that explores the potential of quantum computing methods in (computational) geometry & topology, which is a very fundamental, yet having increasingly significant impacts to real-world problems. What else can stem from this work is a completely interesting avenue.

---

[1] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[2] 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.

[3] Seth Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996.

[4] 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.

[5] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

[6] Andrew M Childs, Robin Kothari, and Rolando D Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017.

[7] Dorit Aharonov, Vaughan Jones, and Zeph Landau. A polynomial quantum algorithm for approximating the jones polynomial. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 427–436, 2006.

[8] Maria Schuld and Francesco Petruccione. *Supervised learning with quantum computers*, volume 17. Springer, 2018.

[9] Nathan Killoran, Thomas R Bromley, Juan Miguel Arrazola, Maria Schuld, Nicolás Quesada, and Seth Lloyd. Continuous-variable quantum neural networks. *Physical Review Research*, 1(3):033063, 2019.

[10] Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

[11] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503, 2014.

[12] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*, 2013.

[13] Nathan Wiebe, Daniel Braun, and Seth Lloyd. Quantum algorithm for data fitting. *Physical review letters*, 109(5):050505, 2012.

[14] Sergey Bravyi, David Gosset, and Robert König. Quantum advantage with shallow circuits. *Science*, 362(6412):308–311, 2018.

[15] Seth Lloyd, Silvano Garnerone, and Paolo Zanardi. Quantum algorithms for topological and geometric analysis of data. *Nature communications*, 7(1):1–7, 2016.

[16] Casper Gyurik, Chris Cade, and Vedran Dunjko. Towards quantum advantage via topological data analysis. *Quantum*, 6:855, 2022.

[17] Ryu Hayakawa. Quantum algorithm for persistent betti numbers and topological data analysis. *Quantum*, 6:873, 2022.

[18] Sam McArdle, András Gilyén, and Mario Berta. A streamlined quantum algorithm for topological data analysis with exponentially fewer qubits. *arXiv preprint arXiv:2209.12887*, 2022.

[19] Shashanka Ubaru, Ismail Yunus Akhalwaya, Mark S Squillante, Kenneth L Clarkson, and Lior Horesh. Quantum topological data analysis with linear depth and exponential speedup. *arXiv preprint arXiv:2108.02811*, 2021.

[20] Andris Ambainis and Nikita Larka. Quantum algorithms for computational geometry problems. *arXiv preprint arXiv:2004.08949*, 2020.

[21] Nhat A Nghiem, Xianfeng David Gu, and Tzu-Chieh Wei. Constant-time quantum algorithm for homology detection in closed curves. *arXiv preprint arXiv:2209.12298*, 2022.

[22] Xianfeng David Gu and Shing-Tung Yau. *Computational conformal geometry*, volume 1. International Press Somerville, MA, 2008.

[23] Ke Ye and Lek-Heng Lim. Schubert varieties and distances between subspaces of different dimensions. *SIAM Journal on Matrix Analysis and Applications*, 37(3):1176–1197, 2016.

[24] Leonard Wossnig, Zhikuan Zhao, and Anupam Prakash. Quantum linear system algorithm for dense matrices. *Physical review letters*, 120(5):050502, 2018.

[25] Zlatko Drmač and Krešimir Veselić. New fast and accurate jacobi svd algorithm. i. *SIAM Journal on matrix analysis and applications*, 29(4):1322–1342, 2008.

[26] Patrick Rebentrost, Adrian Steffens, Iman Marvian, and Seth Lloyd. Quantum singular-value decomposition of nonsparse low-rank matrices. *Physical review A*, 97(1):012327, 2018.

[27] Armando Bellante, Alessandro Luongo, and Stefano Zanero. Quantum algorithms for svd-based data representation and analysis. *Quantum Machine Intelligence*, 4(2):20, 2022.

[28] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 193–204, 2019.

[29] Guang Hao Low and Isaac L Chuang. Optimal hamiltonian simulation by quantum signal processing. *Physical review letters*, 118(1):010501, 2017.

[30] A Yu Kitaev. Quantum measurements and the abelian stabilizer problem. *arXiv preprint quant-ph/9511026*, 1995.

[31] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[32] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, 2014.

[33] James M Varah. A lower bound for the smallest singular value of a matrix. *Linear Algebra and its applications*, 11(1):3–5, 1975.

[34] Nhat A Nghiem and Tzu-Chieh Wei. Quantum algorithm for estimating eigenvalue. *arXiv preprint arXiv:2211.06179*, 2022.

[35] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. *arXiv preprint arXiv:1603.08675*, 2016.

[36] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 217–228, 2019.

Appendix A: Proof of Lemma 4

Here we provide the proof of Lemma 4. To remind, we have:

$$\tilde{p}_0 = \frac{4}{\pi^2 k} \sum_{i=1}^{k} (\arccos \sqrt{\tilde{\lambda}_i})^2, \tag{A1}$$

$$p_0 = \frac{4}{\pi^2 k} \sum_{i=1}^{k} (\arccos \sqrt{\lambda_i})^2. \tag{A2}$$

We also have the following:

$$\sqrt{\frac{4}{\pi^2 k} \sum_{i=1}^{k} \left(\arccos(\sqrt{\tilde{\lambda}_i}) - \arccos(\sqrt{\lambda_i}) \right)^2} \leq \epsilon_p. \tag{A3}$$

For simplicity, let $\Delta = \tilde{p}_0 - p_0$, $\arccos \sqrt{\tilde{\lambda}_i} = \tilde{x}_i$ and $\arccos \sqrt{\lambda_i} = x_i$. We have:

$$\Delta^2 = (\tilde{p}_0 - p_0)^2 \tag{A4}$$

$$= \frac{4^2}{\pi^4 k^2} \left(\sum_{i=1}^{k} (\tilde{x}_i^2 - x_i^2) \right)^2. \tag{A5}$$

Now we notice the following: for arbitrary two real numbers x and y, there always exists a number D such that $(x - y) \leq D(x + y)$. Apply this observation, and we have, for all i: $\tilde{x}_i^2 - x_i^2 = (\tilde{x}_i - x_i)(\tilde{x}_i + x_i) \leq D_i (\tilde{x}_i - x_i)^2$. If we denote $D \equiv \max_i \{D_i\}_{i=1}^{k}$ then we have:

$$\Delta^2 \leq \frac{4^2}{\pi^4 k^2} \left(\sum_{i=1}^{k} D_i (\tilde{x}_i - x_i)^2 \right)^2 \leq \frac{4^2}{\pi^4 k^2} \left(\sum_{i=1}^{k} D(\tilde{x}_i - x_i)^2 \right)^2. \tag{A6}$$

Taking the square root of both sides yields:

$$|\Delta| \leq D \frac{4}{\pi^2 k} \sum_{i=1}^{k} (\tilde{x}_i - x_i)^2, \tag{A7}$$

$$\leq D\epsilon_p = \mathcal{O}(\epsilon_p), \tag{A8}$$

where the last line comes from Eqn. A3. Note that $|\Delta|$ is exactly $|\tilde{p}_0 - p_0|$, therefore, our proof is completed.

Appendix B: Block Encoding From Memory Model

Here we explicitly show that the memory model 7 naturally encodes a unitary block. We remark that it has been mentioned in the original work [24] as well.

According to Lemma 7, we have the following unitary:

$$U_M |k\rangle |0\rangle = \frac{1}{||A_i||} \sum_{i=1}^{n} A_{ik} |k\rangle |i\rangle, \tag{B1}$$

$$U_N |0\rangle |j\rangle = \frac{1}{||A||_F} \sum_{i=1}^{m} ||A_i|| |i\rangle |j\rangle. \tag{B2}$$

We observe that:

$$\langle 0, j| U_N^\dagger U_M |k\rangle |0\rangle = \frac{A_{jk}}{|A|_F}, \tag{B3}$$

which matches perfectly with the definition of the block encoding 1. Therefore, this is a very useful property of the memory model, making it easier to incorporate with quantum signal processing to construct a quantum algorithm, as outlined in the paper.