Quantum Algorithm for Principal Component Analysis

Riccardo Santambrogio, Michele Scandelli, Daniele Tagliabue

August 2020

Abstract

As new milestones are reached in quantum computing and the technology advances towards maturity, an important and promising field of application is that of Quantum Machine Learning. Quantum implementation of subroutines opens up new possibilities for computational speed-up in many algorithms. In this work we analyze one such algorithm in particular, namely Principal Component Analysis. We briefly describe the steps of the classical algorithm to understand how it can be tackled through quantum techniques, which techniques have been proposed and how they work. Hence, a Python implementation of Quantum PCA (QPCA) is provided using Qiskit. We analyze the complexity of the algorithm and show its results in both the simulator and one of IBM's real quantum devices.

Introduction

Principal Component Analysis (PCA) is a mathematical technique that is widely used in machine learning for applications such as dimensionality reduction and lossy data compression. A common interpretation of PCA is that of orthogonal projection of a dataset onto a lower dimensional linear space (the *principal subspace*) such that the variance of the projected data is maximized [1]. Given a d dimensional dataset the algorithm builds d new features as linear combinations of the original ones and ranks them by the portion of variance in the data that they capture. By retaining the top k of these new features one obtains the lower dimensional space that is optimal in terms of explained variance.
The way in which this is done by the algorithm is by working on the covariance matrix of the standardized data. The eigenvectors of this matrix are the new candidate features and the corresponding eigenvalues are proportional to the amount of captured variance. If the covariance matrix was normalized with respect to its trace, then the eigenvalues express directly a percentage.
So performing eigendecomposition of the covariance matrix is enough to determine the principal subspace. It should be noted immediately that this technique proves most useful when only a small number of principal components can explain most of the variance in the data, *i.e.* when many eigenvalues are close to

zero or small, as we are going to use this later.
The most expensive steps of the algorithm are the computation of the covariance matrix and the eigendecomposition. We only address the latter, and use quantum methods proposed in [2, 3, 4] to find eigenvalues and eigenvectors of a real-valued positive semi-definite matrix, such as a covariance matrix.
In particular, this work investigates in the direction of [4], which proposes a slightly modified version of the original algorithm of [3], and explores the possibilities of a more general algorithm for QPCA that uses the currently available frameworks.

Quantum Algorithm

The main idea of the algorithm is provided in [3] and consists in applying Quantum Phase Estimation (QPE) to reveal information about eigenvalues and eigenvectors of the matrix.
Quantum Phase Estimation is a core technique used in many quantum algorithms to estimate the phase of an eigenvector of a unitary operator, *i.e.* given a unitary operator U and eigenstate $|\psi\rangle$ such that

$$U|\psi\rangle = e^{2\pi i\varphi}|\psi\rangle$$

the QPE estimates φ.
In order to apply this to a covariance matrix X we consider $U = e^{iX}$. X is normalized w.r.t. its trace, so that U is unitary, and $2\pi\varphi$ is eigenvalue of X corresponding to eigenvector $|\psi\rangle$.
A description of the procedure, motivated by an application in quantum chemistry for the estimation of energy levels of Hamiltonians, is found in [5].
The circuit for the QPE uses first Phase Kickback to encode φ into an m qubits register, in the form

$$|\Phi\rangle = \frac{1}{\sqrt{2^m}}\sum_{y=0}^{2^m-1} e^{2\pi i\varphi y}|y\rangle \tag{1}$$

This step requires that e^{iX} be implemented as a (controlled) quantum gate; this nontrivial operation is discussed later but for now we assume it can be done efficiently.
After the "phase state" in Eq. (1) has been synthesized, φ is obtained by applying the Inverse Quantum Fourier Transform (IQFT) to it. The m qubits of the original register collapse to binary values that reveal the binary fraction representation of φ. Since φ is in general a real number, for irrational values the procedure yields an approximate answer over m bits.

$$\varphi \approx 0.x_1x_2...x_{m-1}x_m$$

The complete circuit that implements the QPE is shown in Fig.1.

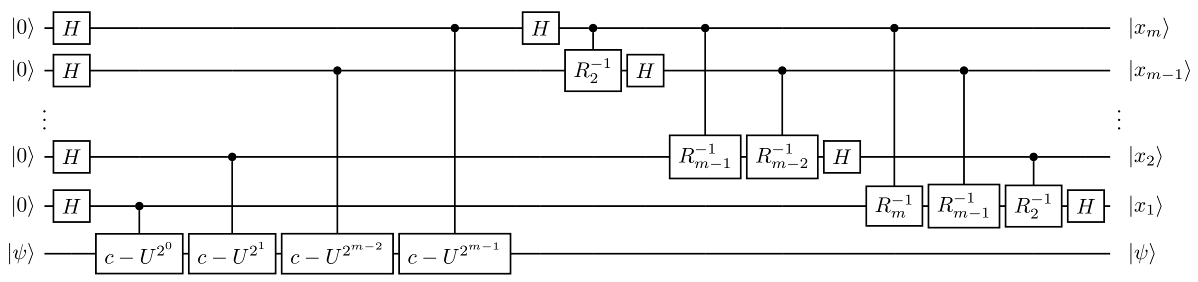

Figure 1: Circuit implementation of Quantum Phase Estimation. $R_k^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{pmatrix}$

While in many applications this algorithm is used to find the eigenvalue corresponding to a known eigenvector, in our case we do not know any of the eigenvectors of X a priori and therefore we cannot prepare the state $|\psi\rangle$.
The intuition [2] that allows to still make use of this, is that, labelling the eigenvectors of X with $|\phi_k\rangle$ and the corresponding eigenvalues with λ_k, we can write any vector $|V\rangle$ (in the same space as the $|\phi_k\rangle$) as a linear combination of $|\phi_k\rangle$ with coefficients c_k

$$|V\rangle = \sum_k c_k |\phi_k\rangle$$

Then, applying Quantum Phase Estimation to state $|V\rangle$ yields measurements of each eigenvalue λ_k with probability $|c_k|^2$, while making this state collapse to the corresponding $|\phi_k\rangle$.
As a consequence, we can say that applying the QPE to any random vector $|V\rangle$ will in general reveal information on all the eigenvalues and eigenvectors of X.
If the matrix X is available in quantum form, *i.e.* a quantum state with density matrix X, using X itself as initial state yields the state

$$\sum_k \lambda_k |\phi_k\rangle \langle\phi_k| \otimes |\lambda_k\rangle \langle\lambda_k|$$

If X is well approximated by a low rank matrix, meaning that few eigenvectors can explain most of the variance in the original data [1], this state will be dominated by few eigenvectors with largest eigenvalues and sampling from it allows to determine the low rank principal subspace [3].
Even if a state X is not available, the procedure can be applied to random initial states, so that the probability that there exists a $c_k = 0$ is zero, and the results can be sequentially improved by using each time the obtained approximation of an eigenvector as new initial state [4].

[1]to see this, consider that the trace of X is dominated by a few large eigenvalues

Analysis

The described algorithm shows that different approaches are possible in exploiting Quantum Phase Estimation to obtain information on the eigendecomposition of a positive semi-definite matrix. Of course, depending on the context of application, some may be more viable than others.
While the most elegant solution is certainly that of using the state X itself to efficiently find the principal components, any eigenvalue-eigenvector couple can be obtained in a polynomial number of trials using a vector for which the corresponding c_k is not exponentially small w.r.t. the size of the problem [2]. Multiple initializations with states generated randomly, or using some criteria, enhance the probability of all outputs being observed a significant number of times.
This is important because, clearly, the quality of the results obtained is determined by our sampling of the final state. So far when we referred to states encoding eigenvectors, that was by way of amplitude encoding, *i.e.* given a vector $\mathbf{x} = \begin{pmatrix} x_0 & x_1 & \dots & x_n \end{pmatrix}^\top$ it is encoded using $\log_2 n$ qubits so that

$$|\mathbf{x}\rangle = x_0 |0\dots00\rangle + x_1 |0\dots01\rangle + \dots + x_n |1\dots11\rangle$$

Since in the final state of the algorithm the eigenvalues and the eigenvectors are entangled, when a λ_k is measured the corresponding $|\phi_k\rangle$ can be sampled; reconstructing this state corresponds to determining the vector's components. The process of reconstructing a quantum state through measurements is called Quantum State Tomography, and is in general computationally intensive: $\mathcal{O}(4^b)$ different observables must be measured for a b-qubit system, and assigning a physically valid density matrix from the gathered data requires an optimization procedure [6]. However, linear scaling can be achieved through statistically based methods when an approximate result is sufficient [7] or when full tomography can be avoided as only partial knowledge of the system is required.
In our implementation, a simple method was adopted that only measures the system in the computational Z basis, to estimate the amplitudes absolute values, and in the "one-X basis", *i.e.* one qubit at a time in the X basis, to estimate the relative phases. With data gathered in these $b + 1$ settings, only $\mathcal{O}(2^b)$ square root and comparison operations are required to obtain an approximation of the state vector, with the implicit assumption of working with real amplitudes[2]. Note that this is linear in the Hilbert space dimension, which corresponds to the dimension of the eigenvectors.
Another crucial part of the algorithm is the implementation of unitary $U = e^{iX}$, as required to perform the QPE. The problem of finding a circuit that approximates such unitary operator for a given $2^n \times 2^n$ hermitian matrix X has also been widely discussed in the literature under the name of Hamiltonian Simulation.
A method for the exponentiation of non-sparse d-dimensional Hamiltonians in

[2]In the ideal execution of the algorithm no rotations around the Y axis are performed, and when an initial state with real components is used also the final state has amplitudes with null imaginary parts.

time $\mathcal{O}(\log d)$ is proposed in [3], and compares to methods based on the higher order Suzuki-Trotter expansion that take $\mathcal{O}(d \log d)$. However, this technique requires once again that many copies of quantum state X are given, and is most effective when some of the eigenvalues of X are large. It was proven that this method is optimal in terms of sample complexity [8], meaning it shows the best bound on the approximation error for a given number of copies of X.
Other methods using different and more convenient input models have been proposed, like the one of [9] based on the qRAM model and with time complexity $\tilde{\mathcal{O}}(\sqrt{d})$.
Finally, it should be pointed out that different algorithms for QPE have been developed, some that are faster than the one presented here which is based on the IQFT. In the circuit of Fig.1, the gate (time) complexity is $\mathcal{O}(m^2)$ for the IQFT plus m Hadamards and calls to powers of $c - U$. If $c - U^k$ is implemented as k calls to $c - U$, then the complexity is $\mathcal{O}(m^2 + 2^m)$. In this case, the complexity scales as the accuracy of the result, which is linear in $M = 2^m$ [2].

Implementation and Results

The algorithm was implemented in Python with Qiskit, using the described circuit for the phase estimation. For the realization of the $c - U$ gates instead the complex matter of Hamiltonian simulation was here avoided, and matrix exponentiation was performed simply using the `expm` function of NumPy [10].
Among the presented approaches for execution, the adopted one was that of preparing random initial states for the circuit, similar to what is done in [4]. The original procedure of [3] was not followed as it requires that the input matrix be encoded into a quantum state. It should be noted that a density matrix that represents a covariance matrix (normalized w.r.t. its trace) corresponds in general to a mixed quantum state. In our case, such a state is not the result of previous quantum computations, and the framework provided by Qiskit does not support a straightforward way to prepare mixed quantum states from scratch.
A demonstration that helps in visualizing how our implementation of the algorithm works is provided in the form of a Jupyter Notebook at [11].
In order to test the algorithm over several input instances, a set of 2000 positive semi-definite matrices was generated following the simple method explained by [12] to obtain a matrix with few large eigenvalues.
We tested the algorithm in Qiskit's QASM simulator for noisy circuits, using multiple random initial vectors and merging the resulting measurements. Thus a single data structure is obtained that contains samples of the eigenvector states grouped by corresponding measured eigenvalues.
In [4] the authors suggest a way to identify the largest eigenvalue, starting from an assumption that the input matrix is well approximated by a matrix with lower rank r and projecting the eigenvalue component on the state corresponding to $1/r$. However, our implementation adopted a different approach that attempts to identify all the eigenvalues appearing in the output state.

Note that, due to noise simulation, a small portion of measurements may yield values that do not correspond to any of the real eigenvalues of the input matrix. Conversely, an eigenvalue with high "quantization error" in our m bit representation may lead to substantial measurements of its different approximations, all entangled with the same eigenvector. These problems, which complicate the automation of the procedure, were addressed by only considering values whose detections are above a threshold percentage and then merging data of values that differ by a single quantization step size. A suitable value for the threshold was experimentally found to be around 10% of the maximum detections for a single value.
Since we are interested in the principal components of the input matrix, we evaluated the results on the *top-k* eigenvalues found. This metric was computed by matching the measured eigenvalues with the real ones (determined by classical computation) and counting the number of subsequent largest eigenvalues found, until the first missing one.
Fig. 2 shows the results obtained for matrices of different size and with different number of random initializations. The MSE of the estimated unit eigenvectors in these settings are provided in Table 1.
As can be seen, the efficacy starts degrading for 16×16 matrices; a reason for this lies in the fact that when the number of eigenvalues grows, they will fall closer to one another, increasing the risk that they might not be distinguishable at the given precision level. This is particularly true for the majority of eigenvalues which are small and get registered as zero or near-zero values.
This being the case, simply increasing the number of precision qubits causes the results to improve significantly (Fig. 3). However, this also comes at a significant cost in circuit complexity, as discussed previously.

Size	2×2	4×4	8×8	16×16
MSE	1.1839×10^{-3}	4.7082×10^{-3}	2.5619×10^{-2}	4.9530×10^{-2}

Table 1

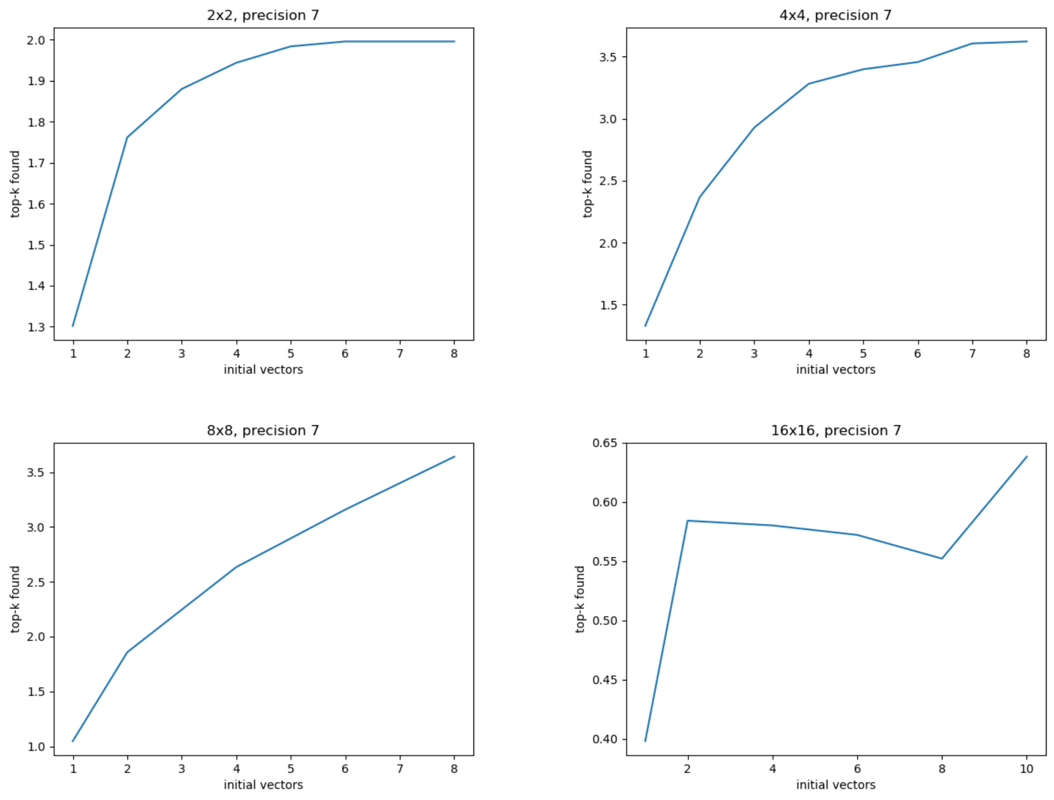

Figure 2: Results for 2×2, 4×4, 8×8 and 16×16 matrices with varying number of initial vectors, each averaged over 500 input matrices drawn without replacement from the 2000 generated. Eigenvalue estimation used 7 qubits precision, each run measured 8192 shots.

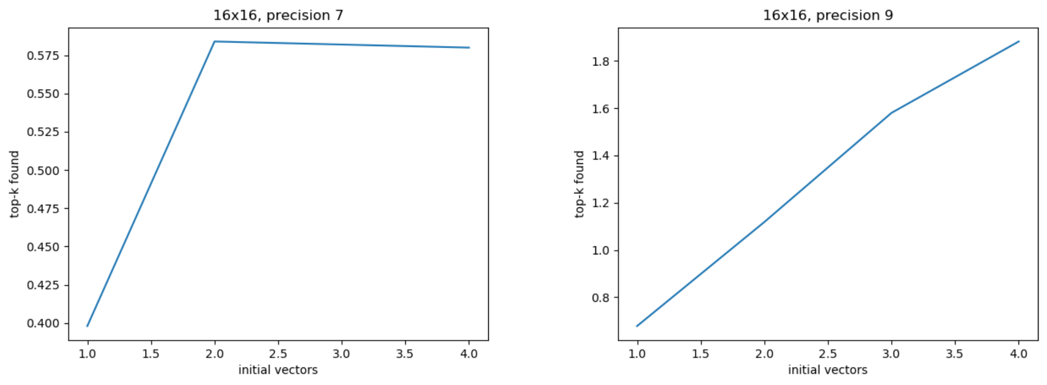

Figure 3: Comparison between 7 and 9 qubit precision for 16×16 matrices

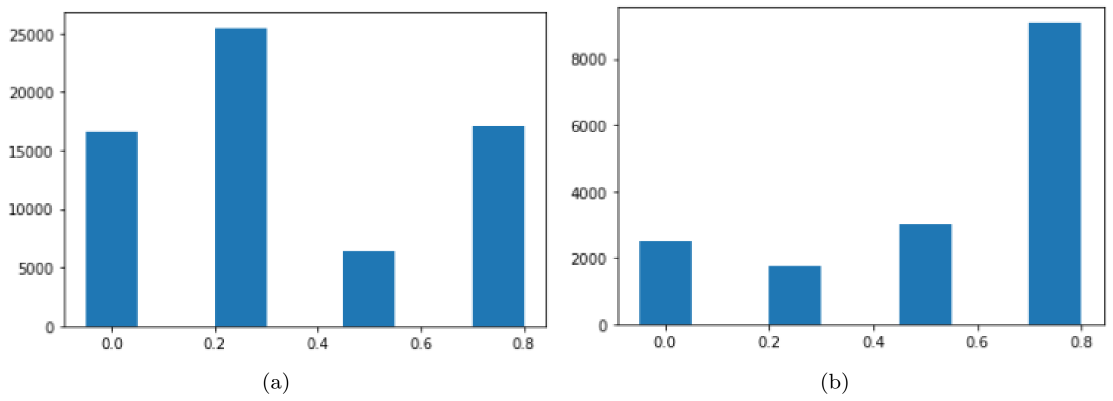

Figure 4: Histogram of measured eigenvalues in the first (a) and second (b) iteration.

The algorithm was then tested on IBM's 5-qubit quantum device Vigo. For this purpose, a 2×2 covariance matrix was used whose eigendecomposition is

$$\lambda_1 = 0.84212698 \quad v_1 = \begin{bmatrix} 0.91854462 & -0.39531732 \end{bmatrix}$$
$$\lambda_2 = 0.15787302 \quad v_2 = \begin{bmatrix} 0.39531732 & 0.91854462 \end{bmatrix}$$

Collecting data from 4 random initial vectors resulted in the histogram shown in Fig. 4a, where corresponding to value 0.75 was the estimated eigenvector

$$v_a = \begin{bmatrix} 0.85450074 & -0.51945017 \end{bmatrix}$$

This reconstructed vector was then used as new initial state for a second iteration of the algorithm, which lead to the histogram in Fig. 4b. Now the eigenvector estimated for value 0.75 was

$$v_{a'} = \begin{bmatrix} 0.91533142 & -0.40270137 \end{bmatrix}$$

This improved result is satisfactory as $v_{a'} \approx v_1$, so the procedure was stopped. If this were not the case, we could keep iterating in this fashion.
The same was attempted for a 4×4 matrix, with the results shown in Fig.5. Clearly, no useful information can be extracted from such noisy outputs, especially in a practical scenario where the real eigendecomposition is not provided as reference.
For larger instance sizes errors arise mainly due to quantum decoherence caused by the depth of the circuit. In moving from the 2×2 to the 4×4 cases we increased the precision from 2 to 3 qubits, as can be seen from the histogram; this is necessary if we hope to identify even just the first two principal components. Nevertheless, with larger input matrices the circuit complexity increases regardless of precision level, due to the *transpilation* process that decomposes larger $c - U$ gates, acting on multiple target qubits, into two-qubit gates that can be physically implemented. Using for instance 2 qubits precision, while a 2×2 input matrix generated a circuit with 30 gates, a 4×4 lead to a 211 gates circuit.

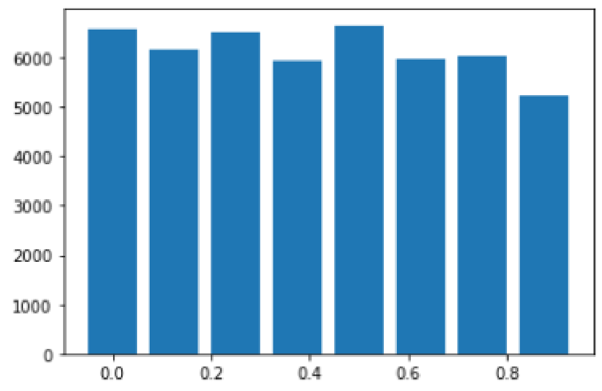

Figure 5: Histogram of measured eigenvalue for a 4×4 matrix

Conclusions

We analyzed the algorithm for QPCA proposed by [3], which has been cited among the most relevant algorithms for quantum machine learning, reviewed its theoretical basis and critical aspects of complexity. We then proposed an implementation that resembles more the modified version of [4], while attempting to tackle the problem of automatizing the reconstruction of the complete eigendecomposition.
Executing the circuit with different random initial states and combining the results, as was done, can be regarded as applying phase estimation to a random mixed quantum state, in consistency with its definition of mixture of pure states. It is worth noting that the same approach could not be employed for the preparation of a mixed state encoding the input matrix, since that would require decomposing such state to find appropriate pure ones, which is commonly done precisely through eigendecomposition.
Testing in the simulator showed expected results, as the algorithm was able to find the principal components of the input matrices starting from different random vectors. A more precise outline of its performance on larger inputs could be achieved by more extensive testing, which however quickly becomes hardly feasible due to the exponential complexity of simulation.
On the real quantum computer, the algorithm behaves as expected for small instances, as demonstrated for a 2×2 matrix. In this case the iterative approach proved most effective, as noted already by [4]. However, errors caused by quantum decoherence and noise led to unusable results already for 4×4 matrices, which by the way almost exhaust the computational resources of currently available quantum computers.
Apart from this, some issues are present with the simple solutions proposed here to fully automatize the procedure and completely remove human supervision, as was required for the testing we performed. These solutions are in general not optimal and might even hurt the performance in some cases. For instance, while merging measurements of close values that correspond to the same true eigenvalue is often necessary for the correct eigendecomposition, this operation becomes harmful when a low precision for estimation causes different true eigenvalues that were found by the algorithm to be mixed together. It is possible that with future error correcting codes these problems could be solved at their root, but currently they must be taken into account as issues that can disrupt the performance of the algorithm.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[2] Daniel S. Abrams and Seth Lloyd. "Quantum Algorithm Providing Exponential Speed Increase for Finding Eigenvalues and Eigenvectors". In: *Physical Review Letters* 83.24 (Dec. 1999), pp. 5162–5165. ISSN: 1079-7114. DOI: `10.1103/physrevlett.83.5162`. URL: `http://dx.doi.org/10.1103/PhysRevLett.83.5162`.

[3] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. "Quantum principal component analysis". In: *Nature Physics* 10.9 (July 2014), pp. 631–633. ISSN: 1745-2481. DOI: `10.1038/nphys3029`. URL: `http://dx.doi.org/10.1038/nphys3029`.

[4] Ana Martin et al. *Towards Pricing Financial Derivatives with an IBM Quantum Computer.* 2019. arXiv: `1904.05803 [quant-ph]`.

[5] Colin P. Williams. *Explorations in Quantum Computing.* 2nd. Springer Publishing Company, Incorporated, 2008. ISBN: 184628886X.

[6] J. B. Altepeter, D. F. V. James, and P. G. Kwiat. "Quantum State Tomography". In: *Lecture Notes in Physics.* Vol. 649. 2004. Chap. 4, pp. 113–145.

[7] Scott Aaronson. "The learnability of quantum states". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 463.2088 (Sept. 2007), pp. 3089–3114. ISSN: 1471-2946. DOI: `10.1098/rspa.2007.0113`. URL: `http://dx.doi.org/10.1098/rspa.2007.0113`.

[8] Shelby Kimmel et al. "Hamiltonian simulation with optimal sample complexity". In: *npj Quantum Information* 3.1 (Mar. 2017). ISSN: 2056-6387. DOI: `10.1038/s41534-017-0013-7`. URL: `http://dx.doi.org/10.1038/s41534-017-0013-7`.

[9] Chunhao Wang and Leonard Wossnig. *A quantum algorithm for simulating non-sparse Hamiltonians.* 2018. arXiv: `1803.08273 [quant-ph]`.

[10] Travis E Oliphant. *A guide to NumPy.* Vol. 1. Trelgol Publishing USA, 2006.

[11] Riccardo Santambrogio, Michele Scandelli, and Daniele Tagliabue. *QPCA Demo.* `https://github.com/Askarpour/sw2_quantum_research/blob/master/SantambrogioScandelliTagliabue/QPCA/notebooks/QuantumPCA_demo.ipynb`. 2020.

[12] amoeba (https://stats.stackexchange.com/users/28666/amoeba). *How to generate a large full-rank random correlation matrix with some strong correlations present?* Cross Validated. URL:https://stats.stackexchange.com/q/125020 (version: 2017-04-13). eprint: `https://stats.stackexchange.com/q/125020`. URL: `https://stats.stackexchange.com/q/125020`.