Teaching quantum computing to computer science students: Review of a hands-on quantum circuit simulation practical

1st Florian Krötz
MNM-Team
Ludwig-Maximilians-Universität München
Munich, Germany
florian.kroetz@nm.ifi.lmu.de

2nd Xiao-Ting Michelle To
MNM-Team
Ludwig-Maximilians-Universität München
Munich, Germany
michelle.to@nm.ifi.lmu.de

3rd Korbinian Staudacher
MNM-Team
Ludwig-Maximilians-Universität München
Munich, Germany
staudacher@nm.ifi.lmu.de

4th Dieter Kranzlmüller
MNM-Team
Ludwig-Maximilians-Universität München
Munich, Germany
kranzlmueller@ifi.lmu.de

***Abstract*—We present a practical course targeting graduate students with prior knowledge of the basics of quantum computing. The practical aims to deepen students' understanding of fundamental concepts in quantum computing by implementing quantum circuit simulators. Through hands-on experience, students learn about different methods to simulate quantum computing, including state vectors, density matrices, the stabilizer formalism, and matrix product states.**

By implementing the simulation methods themselves, students develop a more in-depth understanding of fundamental concepts in quantum computing, including superposition, entanglement, and the effects of noise on quantum systems.

This hands-on experience prepares students to do research in the field of quantum computing and equips them with the knowledge and skills necessary to tackle complex research projects in the field. In this work, we describe our teaching approach and the structure of our practical, and we discuss evaluations and lessons learned.

***Index Terms*—Quantum Computing, Quantum Circuit Simulation, Quantum Circuit Simulation Methods, Quantum Education, Graduate Education**

I. Introduction

The growing importance of quantum computing has created a need for skilled computer scientists to work in this field. However, teaching quantum computing to computer science students is a challenging task, as they often lack a fundamental understanding of quantum mechanics, including concepts such as superposition and entanglement.

The target audience for the practical consists of students who have successfully completed an introductory lecture on quantum computing [1]–[4]. The lecture provides a solid foundation in the fundamentals of quantum computing, but students often require additional support to develop a solid understanding of the subject.

To address this knowledge gap, we designed a Quantum Computing Practical, a hands-on learning experience aimed at deepening students' understanding of quantum computing principles. The practical should not replace an introductory lecture; it builds on the lecture. The primary goals are to equip students with a more in-depth understanding of quantum computing to write and do research in the field.

Our practical course is delivered in an intensive one-week block format. This structure encourages a deeper immersion into the quantum simulation topic and allows students to complete more ambitious projects within a condensed timeframe.

The structure of this paper is as follows: Section II gives an overview of existing teaching approaches for computer science students and motivates why we decided for the teaching approach and learning goals described in Section III, while Section IV outlines how we implemented the described approach. In Section V, we present the evaluation of the practical, provide conclusions and an outlook on future work.

II. Related Work

A growing number of educational initiatives have been developed to support teaching quantum computing [5]–[7], often leveraging interactive tools and simulators [8], [9]. Our approach is more aligned with constructivist learning paradigms, where students gain more in-depth understanding by constructing the quantum circuit simulators themselves.

There is a growing number of simulators for quantum computing [10]. One goal of this practical is that students can make a well-informed decision which simulator fits best for their use case.

Several teaching platforms or toolkits have emerged to support quantum computing education. For instance, IBM Quantum Learning [11], Qiskit [12], and Cirq [13] provide cloud-based access to quantum hardware and simulators. However, while these platforms are good for demonstrating quantum algorithms, they often abstract away implementation details.

In contrast, our course emphasizes building simulators from scratch, helping students internalize core computational models such as statevectors [2], density matrices [2], the stabilizer formalism [3], [14], [15], and matrix product states [16], [17].

Some courses and workshops have incorporated tensor network methods into their curriculum, such as the tutorials provided at the Quantum Tensor Network Summer School [9] targeting PhD students, or targeting students with more prior knowledge [8].

Our practical gives a comprehensive introduction to tensor networks for computer science students. Linear algebra lectures for computer science students usually do not introduce tensors [18]–[20]. Some lectures about deep learning do introduce tensors [21], but since these are advanced lectures, not every computer science student attended such a lecture. However, not having the prior knowledge about tensors and tensor networks makes reading literature on the matrix product state [16], [17] harder to understand.

III. Teaching approach

Computer scientists in quantum computing often use simulators to understand and analyze quantum systems. With multiple simulation methods available, it is essential to know the strengths and weaknesses of each method to choose the right simulator and operate it correctly.

A. Pedagogical Concept

The practical course goes beyond teaching simulators, focusing on developing an in-depth understanding of quantum computing fundamentals through hands-on experience by implementing and optimizing various simulation techniques and exploring noise models.

Throughout the course, supervisors have been present to provide guidance and support, ensuring that students have access to expert knowledge and advice whenever needed. Students have also been encouraged to collaborate and help each other.

For the practical, we use the gate model as the foundation for representing quantum circuits. Since the focus of the practical is understanding the fundamentals in quantum computing, we decided not to use an established implementation of the gate model like Qiskit [12], Cirq [13], or OpenQASM [22], because there is a lot of existing software that abstracts away the underlying quantum computing concepts. For example, Qiskit implements many high-level functions where the user does not have to work at the gate level. We want the students to think about quantum computing on the gate level. OpenQASM has many complex features, like registers or loops, that make it harder to parse. Therefore, we developed a simple representation of quantum circuits, inspired by the representation used in [14]. The core idea here is that a quantum circuit can be represented as a list of gates operating on a set of qubits. By using a simple representation, students have to focus on the underlying principles.

B. Learning Goals

By quantum circuit simulation, we mean to determine the output state $|\psi'\rangle$ obtained by applying a unitary operator U, which represents one or more quantum gates, to an initial state $|\psi\rangle$:

$$|\psi'\rangle = U\,|\psi\rangle\,.$$

For all methods, a central learning goal is the understanding how a quantum state $|\psi\rangle$ is represented, what are strength and weaknesses of this representation and how it is modified when applying a unitary U. The primary learning goals for the practical are:

- A more in-depth understanding of superposition and entanglement is built, since students learn about 4 different methods of how to represent a quantum state on a classical computer.
- A solid understanding of the various simulation techniques, enabling students to make informed decisions about which simulator to use.
- An intuition for the non-local nature of quantum computing, which means that quantum circuits are generally not separable and representing a general quantum state is exponential hard.

The main learning goals for the four chosen methods are:

- State Vector: When representing a quantum state as a State vector, the memory needed grows exponential $O(2^n)$, where n is the number of qubits. Single-qubit operations may change every entry of the state vector, illustrating non-locality. Circuits are represented as a list of gates, but every entry in the state vector needs to be touched, demonstrating non-separability because of the non-local nature of quantum mechanics.
- Density Matrix with Noise: When representing a quantum state as a Density Matrix, the memory needed grows exponential $O(2^{2n})$ and needs quadratic more memory than a state vector. An advantage is that noise can be modeled. Another learning goal is what error types are there and how they are modeled.
- Stabilizer Formalism: The main limitation is that only stabilizer states can be represented, but they can be efficient represented with $O(n^2)$. Quantum systems can be simulated efficiently, but their quantum nature alone does not ensure their usefulness.
- Matrix Product State (MPS): In an MPS, the space complexity depends on the degree of entanglement in the system. Students learn how to decompose an entangled system, building a better understanding of entanglement and showing what makes quantum systems hard to simulate.

  A small but worth mentioning learning goal here is that students get an intuition for what tensors are.

IV. Course

A. Course Description

The official description is: ”The topic of the practical course is simulation of quantum circuits. Participating students are learning different methods for classical simulation of quantum circuits. Based on those, they implement and optimize their own simulator in groups of two.”

The course is structured in two stages: There is a worksheet phase where the students work on worksheets. For each simulation method, there is one worksheet. The second phase is the project phase, where the students implement their own simulator. Our practical course is delivered in an intensive seven-day block format. This structure encourages a deeper immersion into the quantum circuit simulation topic and allows students to complete more ambitious projects within a condensed time frame.

B. Circuit representation

The quantum circuit representation used in our practical is called QCP (Quantum Computing Practical). The representation dispenses with advanced features such as multiple quantum registers, classical registers, and conditional gates.

QCP circuits are represented in text files that need to be in the following form. The first line contains the number of qubits. Each of the following lines represents exactly one gate in the format: `gateName [parameter] [controlqubit(s)] targetqubit`. The parameters `controlqubit` and `targetqubit` are natural numbers between 0 and the number of qubits-1; `parameter` is a decimal number, optionally multiplied by a fraction of π.

For example, a two-qubit circuit that produces a Bell state by first applying a Hadamard gate on the first qubit and then a CX gate, with the control qubit being the first qubit and the target qubit the second, is described as follows:

```
2
h 0
cx 0 1
```

We provide the students with a reference implementation in Python and C. The representation is deliberately kept simple such that writing their own parser in another language is not difficult.

C. Worksheets

The worksheet phase of the practical contains six worksheets that the participants are working on. The first worksheet is a recap, and worksheets 2–5 introduce the simulation methods. The time scope is approximately half a day for each worksheet; in sum three days for the worksheet phase.

1) Recap: The first worksheet is a recap where lecture topics important for the practical course are covered. Thereby, the worksheets have a special focus on unitary operations, tensor products, how to define quantum gates in Dirac notation and as matrices, and density matrices.

2) Statevector: This worksheet introduces statevector simulation and familiarizes students with the QCP circuit representation. In the sheet, students implement a not optimized state vector simulator in python.

The first task is to initialize a state vector $|\psi\rangle$ with n qubits in the state $|0\rangle^{\otimes n}$. Here, the students learn that the state vector has to grow exponential with $O(2^n)$. The next task is to implement gates that modify the state vector $|\psi\rangle$. We ask to create a unitary matrix $U = I \otimes \cdots \otimes U_i \otimes \cdots \otimes I$ by multiplying, using the Kronecker product, identity matrices I and U_i being a single-qubit unitary acting on qubit i or a CNOT unitary unitary that is constructed by

$$CNOT = \underbrace{|0\rangle\langle 0|}_{\text{control}} \otimes \underbrace{I}_{\text{target}} + \underbrace{|1\rangle\langle 1|}_{\text{target}} \otimes \underbrace{U}_{\text{target}}.$$

If the control and target qubit are not next to each other Kronecker products of identity matrices have to be placed between the control and target qubit. This matrix is then multiplied on the state $|\psi'\rangle = U|\psi\rangle$.

We ask the students to validate their implementation by using circuits that produce a Bell state or a GHZ state, and we ask them to visualize the result. Through validating their implementation and visualizing the results, the students get a good understanding of how a state vector represents a quantum state.

Finally, the students implement measurements on single qubits using the measurement operators $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$. The measurement operators are applied to the state vector like single-qubit gates. After the measurement operator has been applied to the state vector, the state vector needs to be normalized with

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi| M_m^\dagger M_m |\psi\rangle}}.$$

3) Density Matrix: The structure of this sheet is similar to the previous sheet on statevector simulation. In the sheet, students implement a not-optimized density matrix simulator in python.

First, a quantum state is initialized as a density matrix $\rho = |\psi\rangle\langle\psi|$. For simplicity, we only consider pure states here. Then gates and measurements with $\rho' = U\rho U^\dagger$ are implemented. Here, the same U as in the state vector sheet can be used. Then the following errors are introduced and how they are modeled: bit flip, phase flip, bit phase flip, depolarization, and amplitude damping [2].

4) Stabilizer Formalism: This worksheet introduces simulation using the stabilizer formalism. This is a pen-and-paper-only sheet. This sheet closely follows Lecture 28 of [3]. We start with the definition of stabilizer states. To represent a quantum state using the stabilizer formalism, we first need to introduce stabilizer states. A state $|\psi\rangle$ is called a stabilizer state when $P|\psi\rangle = |\psi\rangle$ for Pauli matrices $P \in \{Z, X, Y, I\}$. Systems with multiple qubits are stabilized by Pauli strings. A Pauli string is a product of Pauli matrices, denoted as $P_1 \otimes \cdots \otimes P_n$, where each $P_i \in P$, $i \in \{1, \ldots, n\}$, is a Pauli matrix. Pauli strings can be efficiently represented in a so-called tableau:

$$\left(\begin{array}{ccc|ccc} x_{1,1} & \cdots & x_{1,n} & z_{1,1} & \cdots & z_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,n} & z_{n,1} & \cdots & z_{n,n} \end{array}\right),$$

P	x_i	z_i
I	0	0
X	1	0
Z	0	1
Y	1	1

The binary matrix on the left is the tableau, and the right table describes how Pauli matrices are encoded in the tableau. The entries in the tableau are binary. A tableau has n rows, with each of them representing a Pauli string. The Pauli strings represented by the tableau stabilize the quantum state. This representation of a state is much denser than a state vector This efficient representation of a state comes with the cost that only stabilizer states can be represented.

When simulating using the stabilizer formalism, only Clifford gates $H, S, CNOT$ can be applied. The Clifford gates can be realized as simple bit operations on a stabilizer tableau [3]. A nice side effect here is that the participants gain a better understanding of the Hadamard gate: "The Hadamard gate swaps the X and Z basis, and in the stabilizer formalism, one just needs to swap the stabilizers for X and Z." [3]

5) Matrix Product State (MPS): The last worksheet before the project sheet introduces matrix product state (MPS) simulation. This is a pen-and-paper-only sheet. Since most computer science curricula do not introduce tensors and the tensor diagram notation, we introduce it with an example most computer science students know: the matrix product. In this example, the matrices A, B and C are rank-2 tensors. The multiplication of

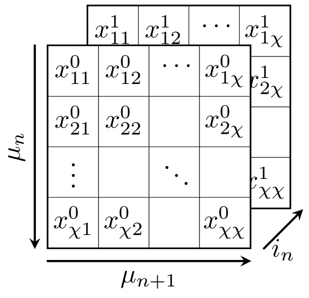
$$C_{i,j} = \sum_k A_{i,k} B_{k,j} \iff \boxed{mmcontraction}$$

is an example of tensor contraction. We then introduce the MPS following [17].

Every qubit is represented by a tensor network of the following form:

$\boxed{tensormps}$

The qubits are ordered in a line, where only adjacent qubits can be entangled. Since most computer science students are struggling to imagine a rank-3 tensors, we give them a visual intuition of how such a tensor in an MPS looks like. These tensors are two matrices:



x^1_{11} x^1_{12} $\cdots$ $x^1_{1\chi}$
$x^1_{2\chi}$
$x^1_{\chi\chi}$
x^0_{11} x^0_{12} $\cdots$ $x^0_{1\chi}$
x^0_{21} x^0_{22} $x^0_{2\chi}$
$\vdots$ $\ddots$
$x^0_{\chi 1}$ $x^0_{\chi 2}$ $x^0_{\chi\chi}$
μ_n
μ_{n+1}
i_n

The front matrix corresponds to the case the qubit is $|0\rangle$, and the back matrix corresponds to $|1\rangle$.

The operation of a single qubit gate is given by

$$M'(n)^{i'_n}_{\mu_{n-1}\mu_n} = \sum_{i_n} U_{i'_n i_n} M(n)^{i_n}_{\mu_{n-1}\mu_n}.$$

In the tensor network diagram, this means plugin the single qubit gate unitary on the open index of the tensor of the corresponding qubit.

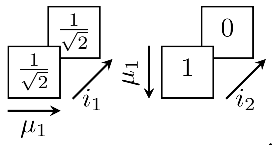
Applying the two-qubit gate in an MPS is more complex. We introduce it using the tensor network diagram:

$\boxed{2qbit2}$

A two-qubit unitary is plugged on the open indices of two adjacent qubits. This two-qubit unitary is represented as a rank-4 tensor. For such a rank-4 tensor, we give the participants the CNOT

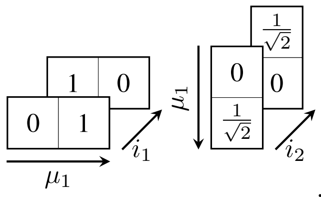
$$CX^{i,j}_{k,l}: \begin{array}{ll} & \overbrace{\quad\quad}^{k=0} \; \overbrace{\quad\quad}^{k=1} \\ CX^{00} = & (1 \quad 0 \quad 0 \quad 0) \\ CX^{01} = & (0 \quad 1 \quad 0 \quad 0) \\ CX^{10} = & (0 \quad 0 \quad 0 \quad 1) \\ CX^{11} = & (0 \quad 0 \quad 1 \quad 0) \\ l= & \;\; 0 \quad 1 \quad 0 \quad 1 \end{array}$$

as an example to build up an understanding of such an operation. After the contraction, the new rank-4 tensor T' needs to be separated into two rank-3 tensors. To obtain two 3-dimensional tensors again, the 4-dimensional tensor is first interpreted as a 2-dimensional matrix (analogous to the 2D representation of the CX gate on the first page of the exercise sheet: Index i'_n and μ_{n-1} form one dimension, index i'_{n+1} and μ'_{n+1} the other). A *singular value decomposition (SVD)* can now be applied to this 2-dimensional matrix. It decomposes a matrix into three components XSY. X and Y are both unitary matrices, and S is a diagonal matrix. The diagonal entries are called *singular values*. By multiplying the singular values to X and interpreting the 2-dimensional matrices as 3-dimensional tensors again (i.e., splitting the X matrix horizontal in the middle into a 0 and a 1 part and the Y matrix vertical), the modified MPS is obtained. The implementation of this procedure with NumPy [23] is trivial. Going through this process with pen and paper illustrates how the simulation process in an MPS works. In our case, the process starts with the state $|0+\rangle$



,

then applying a CNOT gate results in



.

Furthermore, it shows that unentangled states can be represented efficiently, and operations that introduce entanglement make the size of the tensors grow.

D. Student Project

The second stage of the practical is the student project. The students work in groups of two or three. For the project, they choose one of the previously introduced simulation methods and implement it in a programming language of their choice. The goal of the project is not just to implement a simulator, but also that the students get creative and integrate their own ideas into their simulator. For this, we provide them with some initial ideas:

- For all simulation methods: Circuit Preprocessing, Parallelization, Cross-Entropy Benchmarking, ...
- Statevector Simulator: Can you apply a gate to a n-qubit vector without calculating a complete $n \times n$-matrix?
- Density Matrix Simulator: How would you implement noise in gates? Can you reproduce the behavior of real quantum computers using IBM error statistics, for example? You can also use error correction circuits (and modify them if necessary or modify your code) to correct the noise.
- Stabilizer Simulator: Can you omit individual T gate terms? What is the overall error induced by this?
- Matrix Product State Simulator: Can you choose the order of the tensor contractions so that the overall calculation runs as efficiently as possible (contraction plan)? How does the total error behave when cutting using χ? Can you improve the result of the singular value decomposition? (Keyword: Canonical Matrix Product States)

E. Assessment

The assessment consists of two parts: a presentation and a discussion afterward. For the assessment, the students prepare a presentation. The students should answer the following questions in their presentation:

- What simulation method did they choose and why?
- What programming language did they choose and why?
- What optimizations did they implement, and what own ideas did they integrate?

If the students can answer these questions profoundly, can explain their project decisions and give suitable answers to the questions in the discussion, they have proven that they reached the learning goals defined in Section III-B.

V. Evaluation, Lessons Learned, and Outlook

A. Evaluation

The practical has been held twice, in 2023 and 2024. In total, 28 students completed it. Table I shows the popularity of the methods. 2023 and 2024. At the end of each practical course, we handed out an anonymous evaluation form to the students. The students evaluated the practical in both years. The evaluation uses a scale from 1 (very good) to 5 (insufficient). Overall, students rated the practical 1.5 (in 2023 and 2024). The data also suggest that the practical training boosted students' interest in quantum computing, with ratings of 1.5 (2023) and 1.7 (2024) on average. Furthermore, the evaluation results indicate that students had no problem following the content. When asked to assess the clarity of the learning objectives, students rated it 1.8 in 2023 and 1.5 in 2024. The results of the exam show that they do not just have fun but also reach the learning goals. Another goal of the practical is to prepare students to write final theses in the field of quantum computing. We recruited 6 participants from the practical for final theses, and all of them finished successfully.

TABLE I
NUMBER OF TEAMS SELECTING EACH METHOD BY YEAR

	2023	**2024**
State Vector	1	1
Density Matrix	1	4
Stabilizer Formalism	1	1
Matrix Product State	4	2

B. Lessons Learned

Due to historical reasons, a portion of the practical was designed as a challenge, where projects from the second phase competed against each other for the "Best Project" award. The intention behind this challenge was to give the students an incentive to excel in their projects and come up with new and interesting ideas. However, the challenge turned out to be impractical, since it is difficult to design a fair competition that gives all simulation methods an equal chance of winning. For instance, the density matrix method is known to be the slowest one but provides interesting insights into the simulation of noise in quantum computing. Therefore, students who want to win the challenge do not choose this simulation method.

C. Outlook

In the future, we aim to enhance the practical course by incorporating more hands-on experience with real hardware. The Leibniz Supercomputing Center offers access to actual quantum devices [24], which we can leverage to provide students with a more immersive learning experience.

In 2024, some students worked on cross-entropy benchmarking [25] and compared their own simulator with real hardware. It would be great to see students reproducing controversial, discussed results of the Google Sycamore Experiment [26], [27], with real Quantum Computers and their own simulators.

References

[1] M. Homeister, *Quantum Computing verstehen: Grundlagen - Anwendungen - Perspektiven*, ser. Computational Intelligence. Springer Fachmedien Wiesbaden, 2013. [Online]. Available: https://books.google.de/books?id=s1SQkQEACAAJ

[2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[3] S. Aaronson, "Introduction to quantum information science," 2022, accessed: 2025-03-10. [Online]. Available: https://www.scottaaronson.com/qclec.pdf

[4] "Introduction to Quantum Computing," https://www.nm.ifi.lmu.de/teaching/Vorlesungen/2025ss/quantum-computing/, Ludwig Maximilian University of Munich, [Online; accessed 01-April-2025].

[5] A. Mjeda and H. Murray, "Quantum computing education for computer science students: Bridging the gap with layered learning and intuitive analogies," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 03, 2024, pp. 61–70.

[6] G. Carrascal, A. A. del Barrio, and G. Botella, "First experiences of teaching quantum computing," *J. Supercomput.*, vol. 77, no. 3, pp. 2770–2799, Mar. 2021.

[7] G. P. Temporão, T. B. S. Guerreiro, P. S. C. Ripper, and A. M. B. Pavani, "Teaching quantum computing without prerequisites: a case study," in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2022, pp. 673–676.

[8] Zuse Institute Berlin, “Workshop on Tensor Methods for Quantum Simulation 2024,” https://tmqs2024.zib.de/, 06 2024, [Online].

[9] European Tensor Network, “Tensor Network based approaches to Quantum Many-Body Systems,” https://nextcloud.tfk.ph.tum.de/etn/index.php/schools/2024-school/, 05 2024, [Online].

[10] A. J. Gangapuram, A. Läuchli, and C. Hempel, “Benchmarking quantum computer simulation software packages: State vector simulators,” *SciPost Phys. Core*, vol. 7, no. 4, Nov. 2024.

[11] IBM, “IBM Quantum Learning,” https://learning.quantum.ibm.com, 04 2025, [Online].

[12] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, “Quantum computing with Qiskit,” 2024.

[13] C. Developers, *Cirq*. Zenodo, Apr. 2025. [Online]. Available: https://zenodo.org/doi/10.5281/zenodo.4062499

[14] S. Aaronson and D. Gottesman, “Improved simulation of stabilizer circuits,” *Physical Review A*, vol. 70, no. 5, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1103/PhysRevA.70.052328

[15] S. Bravyi, D. Browne, P. Calpin, E. Campbell, D. Gosset, and M. Howard, “Simulation of quantum circuits by low-rank stabilizer decompositions,” *Quantum*, vol. 3, p. 181, Sep. 2019. [Online]. Available: https://doi.org/10.22331/q-2019-09-02-181

[16] G. Vidal, “Efficient classical simulation of slightly entangled quantum computations,” *Phys. Rev. Lett.*, vol. 91, p. 147902, Oct 2003. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.91.147902

[17] Y. Zhou, E. M. Stoudenmire, and X. Waintal, “What limits the simulation of quantum computers?” *Phys. Rev. X*, vol. 10, p. 041038, Nov 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.10.041038

[18] Uni Hamburg, “Lineare Algebra für Informatiker und Statistiker,” https://www.math.uni-hamburg.de/home/belgun/LA-lmu.pdf, 04 2025, [Online].

[19] Kemper, Gregor, “Lineare Algebra für Informatik,” https://www.math.cit.tum.de/fileadmin/w00ccg/algebra/people/kemper/lectureNotes/LA_Inf.pdf, 10 2024, [Online].

[20] Strang, Gilbert, “Linear Algebra,” https://ocw.mit.edu/courses/18-06-linear-algebra-spring-2010/pages/syllabus/, 06 2023, [Online].

[21] quantstart, “Scalars, Vectors, Matrices and Tensors - Linear Algebra for Deep Learning,” https://www.quantstart.com/articles/scalars-vectors-matrices-and-tensors-linear-algebra-for-deep-learning-part-1/, 04 2025, [Online].

[22] A. Cross, A. Javadi-Abhari, T. Alexander, N. De Beaudrap, L. S. Bishop, S. Heidel, C. A. Ryan, P. Sivarajah, J. Smolin, J. M. Gambetta, and B. R. Johnson, “Openqasm 3: A broader and deeper quantum assembly language,” *ACM Transactions on Quantum Computing*, vol. 3, no. 3, Sep. 2022. [Online]. Available: https://doi.org/10.1145/3505636

[23] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[24] Leibniz-Rechenzentrum. (2025) Quantum integration centre des lrz. [Online]. Available: https://www.lrz.de/technologien/quantum/

[25] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, “Characterizing quantum supremacy in near-term devices,” *Nat. Phys.*, vol. 14, no. 6, pp. 595–600, Jun. 2018.

[26] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, “Quantum supremacy using a programmable superconducting processor,” *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019.

[27] F. Pan, K. Chen, and P. Zhang, “Solving the sampling problem of the sycamore quantum circuits,” *Phys. Rev. Lett.*, vol. 129, p. 090502, Aug 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.129.090502