Qudit-Based Quantum State Preparation via KP-Trees*

Simone Ianniciello[1,*], Gianna M. Del Corso[1]

[1]Department of Computer Science, University of Pisa, Italy

Abstract

A KP-Tree is a classical data structure, originally introduced by Kerenidis and Prakash, which stores the squares of the elements of a real vector in $\mathbb{R}^N$ in the leaves of a binary tree while each non-leaf node stores the sum of its children. The KP-Tree has been specifically designed to allow, given quantum access to the levels of the tree in superposition, the preparation of the state in time $O(\text{polylog}(N))$.

This work extends the idea of KP-Trees to enable amplitude encoding of a vector in $\mathbb{R}^N$ on qudit architectures, where a single qudit is represented as a normalized vector in $\mathbb{C}^d$. Two methods are presented to generate the tree and to perform the corresponding state preparation. The first approach utilizes single-parameter Givens rotations to perform the d-level interactions as a sequence of 2-level rotations; the second approach makes use of Householder reflectors to fully utilize the $d-1$ degrees of freedom of a single-qudit gate. From a computational perspective, both approaches achieve a decrease in the number of controls of the quantum gates compared to the original qubit-based algorithm. The approach based on the Householder reflector also demonstrates a reduced computational cost given the reduced tree depth resulting from its higher degrees of freedom.

Keywords

Quantum Computing, Qudit State Preparation, Amplitude Encoding,

1. Introduction

While quantum algorithms are most commonly designed for qubits, there is a growing body of research that designs them directly on qudits, i.e. d-dimensional quantum systems with $d > 2$. The motivation for such interest is both physical and computational. Most quantum hardware is natively multilevel rather than binary, so encoding a qubit amounts to deliberately discarding all but two of the available energy levels [1, 2]. Working on these systems directly as qudits avoids artificial restrictions and has been realized on several experimental platforms: trapped-ion processors have demonstrated universal qudit computation with local dimension up to seven [1, 3], while superconducting platforms have run textbook algorithms on qutrit and transmon-based devices with dedicated gate-synthesis methods now available [4, 5, 6, 7].

The larger state space of a single qudit reduces circuit complexity since an n-qudit register spans a Hilbert space of size d^n, encoding the same information with fewer physical elements. Qudits also enable more efficient compilation of multi-controlled operations and shallower circuits for a range of algorithms [2].

Amplitude encoding is a key primitive for a range of quantum algorithms, including quantum linear solvers [8], recommendation systems [9], and quantum machine learning [10, 11]. With amplitude encoding, the elements of a normalized vector $|\psi\rangle \in \mathbb{C}^N$ are stored in the amplitudes of a quantum register of $n = \lceil \log_2(N) \rceil$ qubits. Kerenidis and Prakash [9] introduced a binary tree classical data structure, the KP-Tree, which stores the squared amplitudes at its leaves and partial sums at internal nodes, enabling state preparation in $O(\text{polylog}(N))$ gates given quantum access to the tree in superposition.

In this paper we generalize the KP-Tree to qudit registers. Replacing the binary tree with a d-ary one reduces its depth from $\lceil \log_2(N) \rceil$ to $\lceil \log_d(N) \rceil$, so that a vector of length N is loaded into a register of $n = \lceil \log_d(N) \rceil$ qudits. The multi-controlled loading gates consequently act on fewer registers,

ICTCS 2026: Italian Conference on Theoretical Computer Science, September 7–9, 2026, Udine, Italy

*Corresponding author.

s.ianniciello@studenti.unipi.it (S. Ianniciello); gianna.delcorso@unipi.it (G. M. Del Corso)

https://simone.ianniciello.me (S. Ianniciello)

0000-0002-6567-0561 (S. Ianniciello); 0000-0002-5651-9368 (G. M. Del Corso)

reducing the number of controls; this structural gain is shared by both methods we propose, differing in how each node's d-dimensional rotation is implemented. The first method emulates the original qubit algorithm within the qudit space: each d-way branching of the tree is realized by a binary subtree of $d-1$ Givens rotations (two level $Ry^{(i,j)}$ gates acting on subspaces i and j of the d-level register) of depth $\lceil \log_2 d \rceil$. The path along the d-ary tree determines the controls of these gates, so the number of controls is reduced with respect to the qubit-based algorithm, while the operations depth stays of order $\log_2 N$.

The second method instead loads the d children's amplitudes at each node with a single Householder reflection, exploiting all the $d-1$ degree of freedom of a single qudit gate to prepare the corresponding node state at once. The reflections are applied level by level, the one at level $t+1$ being controlled on the path encoded in the first t qudits. Because each node is loaded by a single reflection, this method avoids the binary sub-expansion of the first one, so that the number of controlled single-qudit operations follows the d-ary tree depth $\lceil \log_d N \rceil$ rather than $\approx \log_2 N$. Provided the reflection conditioned on the precomputed Householder vector can be applied efficiently, this lowers the overall circuit depth with respect to the qubit-based algorithm.

We release a demo of the presented algorithms in [12], tested with a modified version of the QCLAB Matlab Toolbox [13] supporting qudit simulations.

The remainder of the paper is organized as follows. Section 2 provides the necessary background on qudits and the primitives used throughout. Section 3 introduces the data structure and the algorithm on qubits, along with the Givens-rotation-based generalization. Section 4 describes the Householder-based algorithm. Section 5 concludes with a discussion of extensions and open problems.

2. Notation and Preliminary Results

We organize the necessary background as follows. We first define the notation for quantum states and qudits, and then we present the two primitives used to generalize KP-Tree to qudits.

2.1. Quantum States and Qudits

A qudit is a natural generalization of a qubit to an Hilbert space of dimension $d \geq 2$. We refer to a single qudit with the notation $|\psi\rangle\!\rangle$ to distinguish from the ket-bra notation used to represent qubit. The space $\mathcal{H}_d \cong \mathbb{C}^d$ is spanned by the generalized computational basis states $|i\rangle\!\rangle = \mathbf{e}_i \in \mathbb{N}^{d\times 1}$, for $i = 0, \ldots, d-1$. A general pure state is obtained as a normalized linear combination of the computational basis states, $|\psi\rangle\!\rangle = \sum_{i=0}^{d-1} \alpha_i |i\rangle\!\rangle$, where $\alpha_i \in \mathbb{C}$ and $\sum_{i=0}^{d-1} |\alpha_i|^2 = 1$. An n-qudits register $|\psi\rangle\!\rangle_n$ spans the tensor product space $\mathcal{H}(n,d) \cong \mathbb{C}^{d^n}$, and therefore has higher information density than an n-qubit register.

Universal computation on qudits is possible with a finite gate set that generalizes the qubit set $\{X, Z, H, S, \mathrm{CNOT}\}$ and recovers it at $d = 2$ [14, 2]. We do not need the full set here: the single-qubit primitives underlying our constructions, namely Givens rotations and Householder reflectors (together with a diagonal phase gate), are introduced in the following sections.

2.2. Givens Rotation

A Givens rotation $G^{(i,j)}(\theta, \varphi)$ is defined as a unitary which acts as the identity on every $|k\rangle$ with $k \notin \{i, j\}$, and on the $\{i, j\}$ subspace as:

$$\begin{pmatrix} \cos\theta & -e^{i\varphi}\sin\theta \\ e^{-i\varphi}\sin\theta & \cos\theta \end{pmatrix}. \tag{1}$$

Given a generic unitary $U \in \mathbb{C}^{d\times d}$, we may use QR decomposition and factorize it as $U = (G_1 G_2 \cdots G_l)D$, where each G_i is a Givens rotation and D is a diagonal phases matrix. By concatenating at most $d(d-1)/2$ such rotations we are able to construct any single-qudit unitary [15].

The single-parameter form $\mathrm{Ry}^{(i,j)}(\theta) = G^{(i,j)}(\theta, 0)$ generalizes the qubit $\mathrm{Ry}(\theta)$ gate, and is the only form we use in Section 3.1, since the data loaded are real.

2.3. Householder Reflectors

A Quantum Householder Reflector (QHR) is the single-qudit gate

$$Q_v = I_d - \frac{2}{\|v\|^2} |v\rangle\rangle\langle\langle v| , \quad v \in \mathbb{R}^d.$$

Let W be a unitary operator such that $W |v\rangle\rangle = \|v\| \; |0\rangle\rangle$, then

$$W Q_v W^\dagger = I - 2 |0\rangle\rangle \langle\langle 0| = \mathrm{diag}(-1, 1, \ldots, 1) = P_0,$$

so $Q_v = W^\dagger P_0 W$ where P_0 is a phase gate acting on level 0 that is native in essentially every architecture. Hence, the cost of constructing Q_v reduces to the cost of W, which is a state preparation circuit whose unitary operator has the elements of v on its first row, and admits two complementary approaches. If the architecture provides only two-level Givens rotations (1), W compiles into $d-1$ rotations $\mathrm{Ry}^{(k,k+1)}(\theta_k)$ acting on $\mathrm{span}(|k\rangle , |k+1\rangle)$. The angles are fixed by the tail partial norms of v, $n_k = \sqrt{\sum_{j=k}^{d-1} |v_j|^2}$, which can be precomputed in $O(d)$ by $n_d = 0$ and $n_k = \sqrt{v_k^2 + n_{k+1}^2}$ for $k = d-1, \ldots, 0$, exactly as the partial sums of a KP-Tree (so that $n_0 = \|v\|_2$). The reflector Q_v then costs $2(d-1)$ two-level rotations plus one phase gate, i.e. $O(d)$.

A key property is that, choosing $|v\rangle\rangle = |\phi\rangle\rangle - \frac{\phi_0}{|\phi_0|}\|\phi\| \; |0\rangle\rangle$, the reflector prepares $|\phi\rangle\rangle$ in a single reflection, $Q_v |0\rangle\rangle = \pm |\phi\rangle\rangle$; this is the single-qudit operation applied at each node of our construction. In this regime, however, the reflector is not the cheapest route to state preparation. Preparing an arbitrary single-qudit state $|\psi\rangle$ directly takes only $d-1$ Givens rotations, whereas realizing $Q_v = W^\dagger P_0 W$ with two-level gates costs $2(d-1)$ rotations. The Householder formulation therefore pays off only when Q_v is available as a native primitive: whenever the architecture couples all levels simultaneously, as in d-pod (star) configurations [16], the reflector is produced by a single physical operation, preparing a node in one shot rather than the $d-1$ rotations of the Givens scheme. Equivalently, an arbitrary single-qudit unitary is then synthesized with $O(d)$ reflectors instead of the $O(d^2)$ two-level rotations of the QR decomposition. This is the regime targeted by our Householder-based construction in Section 4.

3. Rotation-based KP-Trees

The original KP-Tree algorithm uses rotations in order to progressively distribute the amplitudes from the root of the tree, corresponding to $|0\rangle$, to their designed position in the quantum state. This process is represented as a binary tree, as each time we progress down a level each partial amplitude gets split between its starting position and a new state. This algorithm is defined by two phases: a classical construction of the tree, and a successive use of the computed elements to form the desired quantum state.

To construct a KP-Tree B for a vector $\psi \in \mathbb{R}^N$ we start by writing the squared norms of the elements of ψ to the leaves of a binary tree of depth $n = \lceil \log_2 N \rceil$, along with their sign. We now fill each internal node v with the sum of its children (i.e. the sum of all the leaves of the subtree rooted in v). Figure 1 shows this construction for a generic vector $|\psi\rangle \in \mathbb{R}^4$.

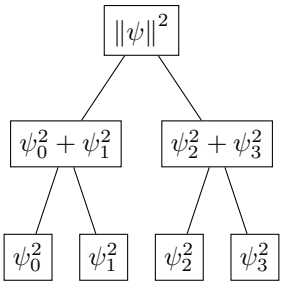
To prepare the state we follow the tree from the root up to the leaves: denoting with a binary string k of length t the index of the k-th node at depth t, we apply rotations on the $t+1$ qubit, conditioned on the first t qubits being in state k as follows

$$|k\rangle |0\rangle \to |k\rangle \frac{1}{\sqrt{B_k}} \left(\sqrt{B_{k0}} |0\rangle + \sqrt{B_{k1}} |1\rangle \right)$$

except for the last step, where we also include the sign in the operator.

On the right of Figure 1 we show how we apply these operators in order to form a generic state $|\psi\rangle \in \mathbb{R}^4$ (omitting the sign for brevity). Assuming a cost of $O(\mathrm{polylog}(N))$ to access the data from the KP-Tree in superposition, the total cost is $O(\mathrm{polylog}(N))$.

Figure 1: left: Default construction of a KP-Tree for $|\psi\rangle$; **right**: Illustration of the operations executed on $|00\rangle$ to obtain the required state: the first operation on qubit 1 depends on the values at depth 1, while the second operation on qubit 2 (conditioned on qubit 1) depends on the children of the previous step

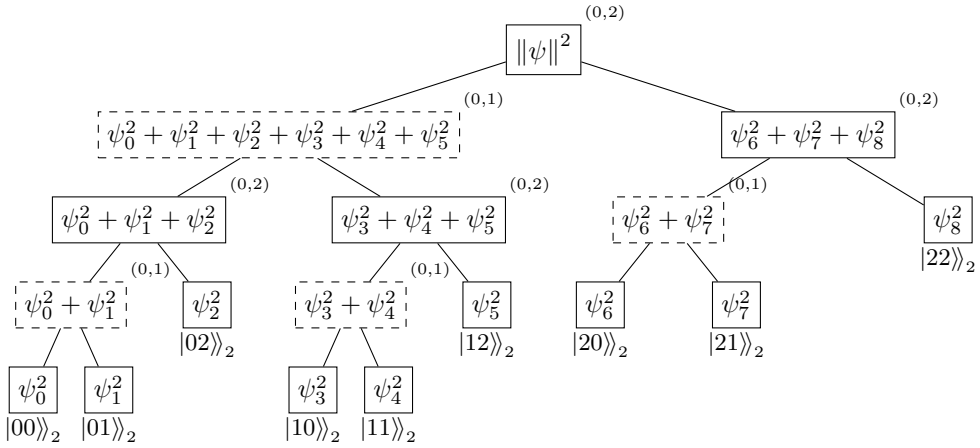


$$
\begin{aligned}
|0\rangle\,|0\rangle &\to \left(\sqrt{\psi_0^2+\psi_1^2}\,|0\rangle + \sqrt{\psi_2^2+\psi_3^2}\,|1\rangle\right)|0\rangle \\
&\to \sqrt{\psi_0^2+\psi_1^2}\,|0\rangle\left(\frac{1}{\sqrt{\psi_0^2+\psi_1^2}}\left(\sqrt{\psi_0^2}\,|0\rangle + \sqrt{\psi_1^2}\,|1\rangle\right)\right) \\
&\quad + \sqrt{\psi_2^2+\psi_3^2}\,|1\rangle\left(\frac{1}{\sqrt{\psi_2^2+\psi_3^2}}\left(\sqrt{\psi_2^2}\,|0\rangle + \sqrt{\psi_3^2}\,|1\rangle\right)\right) \\
&= \sqrt{\psi_0^2}\,|00\rangle + \sqrt{\psi_1^2}\,|01\rangle + \sqrt{\psi_2^2}\,|10\rangle + \sqrt{\psi_3^2}\,|11\rangle
\end{aligned}
$$

3.1. Qudit Generalization via Givens Rotations

To build the QdKP-Tree to prepare a state[1] $|\psi\rangle\!\rangle_n \in \mathbb{R}^N$ with $n = \lceil \log_d N \rceil$ we start by building a d-ary tree, filling the leaves as in the binary algorithm. We then decompose any d-ary subtree of depth 2 into a binary tree, as we are only allowed 1 degree of freedom by each Givens rotation. In particular we substitute them with binary trees of maximum depth $l = \lceil \log_2 d \rceil$. The internal nodes are computed as in the qubit case. Figure 2 shows the resulting tree for a register of two qutrits $|\psi\rangle\!\rangle_2$.

Figure 2: QdKP-Tree construction for $|\psi\rangle\!\rangle_2 \in \mathbb{R}^9$. Dashed nodes have been added to convert the ternary tree into a binary tree. Internal nodes are marked with the subspace on which the generalized $\mathrm{Ry}^{(i,j)}$ gate operates.



The first step to prepare the state defined by this example is as follows:

$$
|0\rangle\!\rangle\,|0\rangle\!\rangle \xrightarrow{\mathrm{Ry}^{(0,2)}} \left(\sqrt{\psi_0^2+\psi_1^2+\psi_2^2+\psi_3^2+\psi_4^2+\psi_5^2}\,|0\rangle\!\rangle + \sqrt{\psi_6^2+\psi_7^2+\psi_8^2}\,|2\rangle\!\rangle\right)|0\rangle\!\rangle
$$

By following this procedure, we are emulating the qubit algorithm in the higher qudit space, where the path in the d-ary tree defines the controls of the operators, and each subtree is executed by a set of Givens rotations.

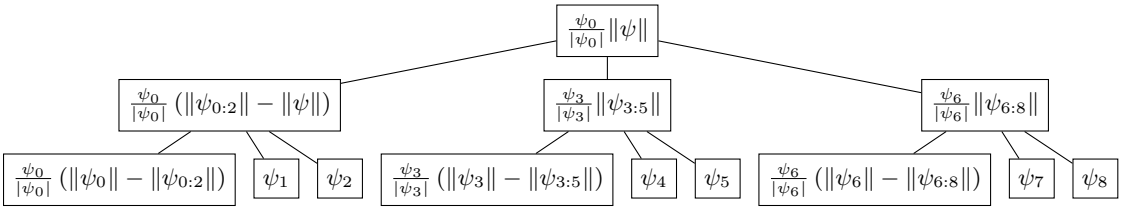
By denoting with N_{mem} the number of gates needed to retrieve data from the tree in superposition i.e. the cost of the quantum memory implementation, the cost for this state preparation algorithm is $O(\log_2(N) \cdot N_{\mathrm{mem}})$ gates. This has the same structure as the qubit case, so any advantage over the qubit case can only stem from faster memory access.

[1]We discuss here only the real case, the extension to complex vectors is too technical for such a short communication.

4. Reflector-based KP-Trees

In this domain, in order to prepare the n qudit state $|\psi\rangle\rangle_n$ we start by generating the d-ary tree with ψ_i on the leaves. The internal nodes are computed as the phase of the left-most leaf in the subtree rooted in it multiplied by the norm of the leaves in the same subtree. We then iterate over the whole tree one more time, replacing the set of children of every node with the corresponding vector $|v\rangle\rangle$ introduced in Section 2.3. In Figure 3 we show the resulting tree for a vector $|\psi\rangle\rangle_2 \in \mathbb{R}^9$.

Figure 3: QdKP-Tree which prepares the state $|\psi\rangle\rangle_2 \in \mathbb{R}^9$ with Householder reflectors.



The value stored at the root of the tree corresponds to a global sign not accounted for by the reflectors; as such the first gate in the circuit is going to be a phase which targets the state $|0\rangle\rangle$ which fixes the correct sign. After that, given k as the d-ary string representing a path on the tree, at each level $t+1$ of the tree we perform the following reflection controlled on the first t qudits being in state k:

$$|k\rangle\rangle_t |0\rangle\rangle \to |k\rangle\rangle_t Q_{v_k} |0\rangle\rangle$$

Below we show the first steps which are performed from the tree on Figure 3

$$\begin{aligned} |0\rangle\rangle |0\rangle\rangle &\xrightarrow{\text{Phase}} \frac{\psi_0}{|\psi_0|} |0\rangle\rangle |0\rangle\rangle \\ \text{QHR on first qutrit} &\to \left(\frac{\psi_0}{|\psi_0|}\|\psi_{0:2}\| \, |0\rangle\rangle + \frac{\psi_3}{|\psi_3|}\|\psi_{3:5}\| \, |1\rangle\rangle + \frac{\psi_6}{|\psi_6|}\|\psi_{6:8}\| \, |2\rangle\rangle \right) |0\rangle\rangle \\ \text{QHR on the second qutrit, conditioned on the first} &\to |\psi\rangle\rangle_2 \end{aligned}$$

Since this method reduces the overall tree depth by a factor of $\log_2(d)$ with respect to the binary tree, the cost for the full algorithm becomes $O\left(\frac{\log(N)}{\log(d)} \cdot N_{\text{mem}}\right)$ yielding a total cost reduction regardless of the memory implementation.

It is worth noting that the algorithm described here can be extended to load complex states with no modifications given a QHR gate which can handle complex reflectors and noting that the root of the tree will contain a complex phase.

5. Conclusions

In this paper we presented two methods for amplitude encoding of a real vector into a quantum register of qudits via a generalization of the KP-Tree data structure. Both methods exploit the d-ary tree structure to reduce the number of controls on the loading gates compared to the original qubit-based algorithm.

The first method is based on Givens rotations and emulates the qubit algorithm in the qudit space, preserving a tree depth of $\lceil \log_2 N \rceil$. The second method, based on Householder reflectors, by utilizing all $d-1$ degrees of freedom of a single-qudit gate, reduces the tree depth to $\lceil \log_d N \rceil$, reducing the gate count when the reflector is available as a primitive gate.

A natural direction for future work is a finer-grained analysis of both methods, decomposing each operation into elementary 1- and 2-qudit gates. A further open question is the cost of implementing quantum access in superposition in the qudit setting, given its importance in these algorithms.

Funding

Partial support is provided by the INdAM - GNCS project "Algebra Lineare Quantistica, State Preparation e Compilazione di Circuiti Quantistici", CUP E53C25002010001. G. Del Corso is also partially supported by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers, Superconducting Quantum Materials and Systems Center (SQMS) under contract number No. 89243024CSC000002.

Declaration on Generative AI

During the preparation of this work, the authors used claude-sonnet-4-6 in order to: Grammar and spelling check. After using these tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] M. Ringbauer, M. Meth, L. Postler, R. Stricker, R. Blatt, P. Schindler, T. Monz, A universal qudit quantum processor with trapped ions, Nature Physics 18 (2022) 1053–1057.

[2] Y. Wang, Z. Hu, B. C. Sanders, S. Kais, Qudits and high-dimensional quantum computing, Frontiers in Physics 8 (2020) 589504.

[3] P. Hrmo, B. Wilhelm, L. Gerster, M. W. van Mourik, M. Huber, R. Blatt, P. Schindler, T. Monz, M. Ringbauer, Native qudit entanglement in a trapped ion quantum processor, Nature Communications 14 (2023) 2242.

[4] T. Roy, Z. Li, E. Kapit, D. I. Schuster, Realization of two-qutrit quantum algorithms on a programmable superconducting processor, Physical Review Applied 19 (2023) 064024. doi:10.1103/PhysRevApplied.19.064024. arXiv:2211.06523.

[5] M. S. Blok, V. V. Ramasesh, T. Schuster, K. O'Brien, J. M. Kreikebaum, D. Dahlen, A. Morvan, B. Yoshida, N. Y. Yao, I. Siddiqi, Quantum information scrambling on a superconducting qutrit processor, Physical Review X 11 (2021) 021010. doi:10.1103/PhysRevX.11.021010.

[6] P. Liu, R. Wang, J.-N. Zhang, Y. Zhang, X. Cai, H. Xu, Z. Li, J. Han, X. Li, G. Xue, W. Liu, L. You, Y. Jin, H. Yu, Performing SU(d) operations and rudimentary algorithms in a superconducting transmon qudit for $d = 3$ and $d = 4$, Physical Review X 13 (2023) 021028. doi:10.1103/PhysRevX.13.021028.

[7] L. E. Fischer, A. Chiesa, F. Tacchino, D. J. Egger, S. Carretta, I. Tavernelli, Universal qudit gate synthesis for transmons, PRX Quantum 4 (2023) 030327. doi:10.1103/PRXQuantum.4.030327.

[8] A. W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for solving linear systems of equations, Physical Review Letters 103 (2009) 150502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.103.150502. doi:10.1103/PhysRevLett.103.150502.

[9] I. Kerenidis, A. Prakash, Quantum Recommendation Systems, 2016. doi:10.48550/arXiv.1603.08675. arXiv:1603.08675.

[10] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, S. Lloyd, Quantum machine learning, Nature 549 (2017) 195–202. URL: https://www.nature.com/articles/nature23474. doi:10.1038/nature23474.

[11] A. Poggiali, A. Berti, A. Bernasconi, G. M. Del Corso, R. Guidotti, Quantum clustering with k-means: A hybrid approach, Theoretical Computer Science 992 (2024) 114466. URL: https://doi.org/10.1016/j.tcs.2024.114466. doi:10.1016/j.tcs.2024.114466.

[12] S. Ianniciello, QdKPTree: Demo implementation, https://github.com/iannisimo/2026-ICTCS-Code, 2026. GitHub repository.

[13] S. Keip, D. Camps, R. V. Beeumen, Quantumcomputinglab/qclab: Qclab v1.1.0, 2025. URL: https://doi.org/10.5281/zenodo.15528792. doi:10.5281/zenodo.15528792.

[14] A. Muthukrishnan, J. Stroud, Carlos R., Multivalued logic gates for quantum computation, Physical Review A 62 (2000) 052309. doi:10.1103/PhysRevA.62.052309.

[15] G. Brennen, D. O'Leary, S. Bullock, Criteria for exact qudit universality, Physical Review A 71 (2005) 052318. doi:10.1103/PhysRevA.71.052318.

[16] P. A. Ivanov, E. S. Kyoseva, N. V. Vitanov, Engineering of arbitrary $u(n)$ transformations by quantum householder reflections, Physical Review A 74 (2006). URL: http://dx.doi.org/10.1103/PhysRevA.74.022323. doi:10.1103/physreva.74.022323.