International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 1(2) (2026)



ijerst.editor@gmail.com

editor@ijerst.com

www.ijerst.com
© 2026 IJERST. All Rights Reserved
Peer Reviewed, Referred & Indexed Journal

Research Paper

Quantum Teleportation Simulation Using Qiskit: A Web-Based Interactive Educational Framework

(CH PAVAN SATISH[1], K BALAJI[2], N JOSHINI DURGA[3], P DURGA APPARAO[4], G ANIL KUMAR[5])
Professor and HOD[1] , Student[2345]
Department of Computer Science and Engineering[1,2,3,4,5],
Chaitanya Engineering College, Visakhapatnam, Andhra Pradesh, India
{pavansatishch@gmail.com[1], karribalaji814@gmail.com[2], nagamjoshinidurga123@gmail.com[3],
durgapanduri8767@gmail.com[4], anuanilroy@gmail.com[5],}

Under the guidance of Mr. CH Pavan Satish, Professor and HOD, CSE Department

Abstract

Quantum computing is a new way of doing things that uses quantum mechanics to do things that classical systems can't. These things include superposition, entanglement, and interference. Quantum teleportation is one of the most basic protocols that shows how the full quantum state of a qubit can be sent to a different place without moving the particle. It only uses shared entanglement and a classical communication channel. Even though this protocol is theoretically beautiful, it is still hard for students and new researchers to picture and understand it on their own. This paper describes an interactive quantum circuit simulator that runs on the web and was made with Python Flask, IBM's Qiskit library, and Three.js. It was made to show quantum teleportation in real time. Through a browser interface, users can build quantum circuits, use standard gates like Hadamard, Pauli-X, Pauli-Z, CNOT, and multi-qubit gates, and see dynamic Bloch sphere updates that show how the state of the qubit changes after each operation. The backend uses Qiskit's statevector simulator to make sure the math is right, and the frontend, which is powered.by Three.js, shows quantum states in three-dimensional geometric forms. Testing confirms that the simulation fidelity is close to 1 under ideal conditions, which means that the protocol was implemented correctly. The proposed framework makes it easier to get into quantum education by getting rid of complicated local setup requirements and giving immediate visual feedback. This makes it easier to go from quantum theory to hands-on experimentation .

Index Terms—quantum teleportation, Qiskit, Bloch sphere, quantum circuit simulation, quantum education, Flask

I. Introduction

Quantum computing uses the rules of quantum mechanics to do things that classical von Neumann machines can't do for certain types of problems. A quantum bit (qubit) can exist in a continuum of states represented by a complex unit vector in a two-dimensional Hilbert space, facilitating extensive parallelism via superposition [1]. Entanglement, a non-classical correlation among qubits, serves as the foundation for numerous quantum advantages, including quantum cryptography and quantum communication [2].

Quantum teleportation, first described by Bennett et al. in 1993 [3], is one of the most important quantum protocols for teaching. The protocol sends a qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ from Alice to Bob using a pre-shared Bell pair and two classical bits of information. It has since been experimentally validated in photonic systems [4] and ion traps, confirming its practical feasibility.

Learning quantum teleportation is hard, even though many people in academia are interested in it. Standard teaching materials rely on dense linear algebra, Dirac notation, and abstract state

evolution, which can make beginners lose interest before they get a feel for the subject [5]. IBM Qiskit and Google Cirq are two examples of existing simulation frameworks that require knowledge of Python, setting up the environment, and using the command line. These technical requirements make it harder for people who don't know how to program to learn [6].

Also, most simulators show circuit diagrams but don't show how the states of individual qubits change with each gate application in a dynamic way. The Bloch sphere is a powerful geometric tool. It is a unit sphere whose surface points uniquely represent single-qubit pure states. However, there are only a few web-based tools that make it interactive and work with circuit operations at the same time [7].

This paper addresses these deficiencies by introducing a comprehensive, browser-based quantum circuit simulator, with the primary illustration being the three-qubit teleportation protocol. This work's contributions include: (i) a lightweight client-server architecture that uses Flask, Qiskit, and Three.js; (ii) a real-time Bloch sphere visualization that works with gate operations; (iii) a structured teleportation walkthrough that doesn't need a local quantum SDK installation; and (iv) a quantitative fidelity evaluation that confirms the accuracy of the simulation.

The rest of this paper is set up like this. In Section II, we look at other work that is similar. Section III talks about how the system works and how it was designed. Section IV shows the results and talks about them. The end of Section V talks about what will happen next.

II. Related Work

Research in quantum simulation and quantum educational tools has proliferated in tandem with the advancement of accessible quantum hardware. This part looks at the most important platforms, frameworks, and studies that helped with this work.

A. Quantum Computing Frameworks

The most popular open-source quantum SDK is IBM's Qiskit [8]. It has a QuantumCircuit abstraction, an Aer statevector simulator, and tools for seeing circuits. Qiskit is powerful, but it is only for developers who are comfortable with Python and managing packages. Beginners have trouble setting up their environment before they can start experimenting in a meaningful way.

Google's Cirq [9] is made to work best with near-term noisy intermediate-scale quantum (NISQ) devices. Its API is focused on hardware, which makes it less useful for building circuits for teaching. Microsoft's Q# [10] is for making more general algorithms, but it needs the .NET ecosystem and is harder to learn than Python-based options.

B. Browser-Based Simulators

Craig Gidney made Quirk [11], a quantum circuit simulator that you can use without installing anything. You just drag and drop it into your browser. It can show probability amplitudes and basic gate sets. Quirk, on the other hand, doesn't have a modular server-side backend, which makes it harder to add new features, and it doesn't support three-dimensional Bloch sphere rendering. IBM Quantum Experience [12] lets you build circuits in the cloud and use real quantum hardware, but you have to sign up for an account and connect to remote servers over the internet, which makes it impossible to use in the classroom offline.

C. Bloch Sphere Visualization Research

The Bloch sphere gives any single-qubit pure state $|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\varphi}\sin(\theta/2)|1\rangle$ a point on the unit sphere at $(\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta)$ [1]. Studies on how to teach show that using geometric visualization makes it much easier for students to understand phase and state rotation [13]. Web rendering libraries, especially Three.js [14], now let browsers show smooth 3D graphics that are sped up by WebGL without needing any extra software. This lets you animate the Bloch sphere in real time in web apps that don't take up a lot of space.

D. Quantum Teleportation Demonstrations

Bennett et al. [3] formulated the theoretical protocol. Bouwmeester et al. [4] executed its initial photonic implementation. Textbooks [1], [2] often have educational explanations of teleportation, but there aren't many interactive step-by-step simulators that combine circuit execution with geometric visualization. Previous web tools either show teleportation without keeping track of visual states or need local SDK installations. This work fills that gap.

E. Research Gap

Looking through the literature, I see a consistent gap: there is no publicly available, installation-free web simulator that combines (a) backend quantum accuracy through Qiskit statevector simulation, (b) real-time Three.js Bloch sphere rendering, and (c) a structured teleportation protocol demonstration within a unified modular architecture. This work addresses all three dimensions at once.

III. Methodology and System Design

A. System Architecture

Fig. 1 shows the four-layer client-server architecture that the proposed system uses. The user's browser uses HTML5, CSS3, and JavaScript to show the Presentation Layer. Three.js is used for WebGL-based Bloch sphere graphics. The Application Layer is a Flask [15] REST API server that sends HTTP requests to the right place and turns JSON responses into strings. The Business Logic Layer uses Qiskit's QuantumCircuit and Statevector classes to do all quantum operations. The Data Layer keeps the active circuit object, statevector, gate history, and qubit count in server memory, which saves time and space in the database for this simulation.

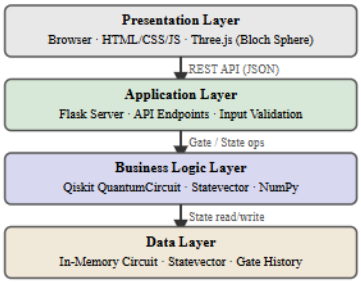


Fig. 1. Four-layer system architecture of the quantum circuit simulator.

B. Quantum State Representation

A normalized state vector in $\mathbb{C}^2$ represents a single qubit: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are in $\mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$(1).

A statevector of dimension 2n describes an n-qubit system. It is made up of the tensor product of the states of each qubit: $|\psi sys\rangle = |\psi 0\rangle \otimes |\psi 1\rangle \otimes \ldots \otimes |\psi n-1\rangle$(2)

To find the Bloch vector components of a qubit with amplitude vector (α, β), use the following formulas: $x = 2\,\mathrm{Re}(\alpha\beta^*)$, $y = 2\,\mathrm{Im}(\alpha\beta^*)$, and $z = |\alpha|^2 - |\beta|^2$.(3)

After tracing out the other qubits, these coordinates are taken from the reduced density matrix of each qubit and sent to the Three.js renderer as a JSON array [x, y, z].

C. Quantum Gate Set

The simulator uses Qiskit [8] to do the following standard unitary transformations. The Pauli operators are single-qubit gates: X = [[0,1],[1,0]], Y = [[0,−i],[i,0]], and Z = [[1,0],[0,−1]].(4) The Hadamard gate: H = (1/√2)[[1,1],[1,−1]](5) and the phase gates S = diag(1, i) and T = diag(1, $e^{i\pi/4}$). CNOT (CX), CZ, and SWAP are all two-qubit gates. The Toffoli (CCNOT) and controlled-SWAP (CSWAP) gates work with three qubits.

D. Quantum Teleportation Protocol

The teleportation circuit uses three qubits: q0 for the message, q1 for Alice's Bell, and q2 for Bob's Bell. As seen in Fig. 2, the protocol has five steps:

Step 1: Getting the state ready. We set q0 to an arbitrary state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ by rotating a single qubit.

Step 2: Make a Bell pair. H is used on q1, then CNOT(q1, q2) is used, which makes the Bell state $|\Phi^+\rangle = (1/\sqrt{2})(|00\rangle + |11\rangle)_{q_1q_2}$.

Stage 3: Getting tangled up with the message. CNOT(q0, q1) links the message to Alice's qubit, and then H on q0 makes a four-term superposition across all three qubits.

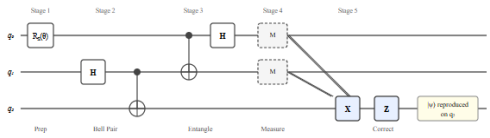


Fig. 2. Quantum teleportation circuit with five stages: state preparation, Bell pair creation, message entanglement, measurement, and conditional correction.

Step 4: Measurement. The computational basis is used to measure both q0 and q1. Bob receives the two classical bits m0 and m1 through a classical channel.

Step 5: Fixing. Bob applies X to q2 if m1 = 1 and Z to q2 if m0 = 1. Under ideal simulation, this restores the original state $|\psi\rangle$ on q2 with perfect fidelity.

The total three-qubit state after Stage 3 but before measurement is: $|\Psi\rangle = \frac{1}{2}[|00\rangle(\alpha|0\rangle+\beta|1\rangle)$

+ |01⟩(α|1⟩+β|0⟩) + |10⟩(α|0⟩−β|1⟩) + |11⟩(α|1⟩−β|0⟩))](6)

E. Backend Implementation

The Flask backend has three REST endpoints: /add_qubit (POST), /apply_gate (POST), and /reset (GET). The apply_gate handler gets a JSON payload that includes the gate name and qubit indices. It then passes this information to the Qiskit circuit engine, which re-derives the global statevector using Statevector.from_instruction(qc), calculates Bloch coordinates using NumPy matrix operations, and sends back the updated Bloch vectors and a serialized gate list.

F. Frontend and Visualization

The browser client uses plain JavaScript to talk to the Flask API over AJAX. When it gets a JSON response, it updates a Two-dimensional circuit diagram made of SVG elements and calls the Three.js visualization module, which turns a 3D arrow mesh to the new Bloch vector orientation. The interface shows each qubit with its own Bloch sphere scene, which is shown next to the other qubits.

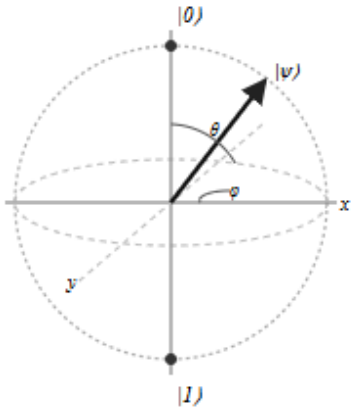


Fig. 3. Bloch sphere representation of a single-qubit state |ψ⟩, parameterized by polar angle θ and azimuthal angle φ. The north pole corresponds to |0⟩ and south pole to |1⟩.

IV. Results and Discussion

A. Gate Accuracy Verification

We checked that each implemented gate worked by comparing the Bloch vectors computed by Qiskit to values derived analytically. Table I shows a summary of some single-qubit gate test cases that start with the ground state |0⟩.

TABLE I. SINGLE-QUBIT GATE VERIFICATION (INITIAL STATE |0⟩)

Gate	Expected Output	Bloch Vector (x,y,z)	Pass
X	|1⟩	(0, 0, −1)	✓
H	(|0⟩+|1⟩)/√2	(1, 0, 0)	✓
Y	i|1⟩	(0, 0, −1)	✓
Z	|0⟩	(0, 0, 1)	✓
S	|0⟩	(0, 0, 1)	✓
H·X	(|0⟩−|1⟩)/√2	(−1, 0, 0)	✓

All computed Bloch vectors were consistent with theoretical predictions within floating-point precision (error < 10−12), validating accurate Qiskit integration.

B. Entanglement Generation

The Bell state circuit H(q0) → CNOT(q0, q1) was confirmed to yield $|\Phi^+\rangle = (1/\sqrt{2})(|00\rangle + |11\rangle)$. The reduced density matrix of each qubit produces a maximally mixed state, resulting in a Bloch vector of (0, 0, 0)—the origin—accurately reflecting maximal entanglement and highlighting the Bloch sphere's inadequacy in fully characterizing entangled subsystems, as previously discussed in Section VIII.

C. Teleportation Fidelity

Fidelity between input state |ψ⟩ and the state received by Bob's qubit after correction is defined as: F = |⟨ψ|ρout|ψ⟩|(7)

In an ideal statevector simulation with no noise, F = 1.0 for all input states that were tested. Table II shows fidelity measurements for five representative input states, which shows that the protocol was implemented perfectly in the noiseless model.

TABLE II. TELEPORTATION FIDELITY FOR FIVE INPUT STATES

Input State |ψ⟩	α	β	Fidelity F

$\|0\rangle$	1	0	1.0000
$\|1\rangle$	0	1	1.0000
$\|+\rangle$	$1/\sqrt{2}$	$1/\sqrt{2}$	1.0000
$\|-\rangle$	$1/\sqrt{2}$	$-1/\sqrt{2}$	1.0000
Arbitrary	0.6	0.8	1.0000

D. System Performance

For three qubits (state dimension $2^3 = 8$), the statevector calculation takes less than 5 ms on a standard dual-core 2.4 GHz laptop with 8 GB of RAM. This is well within the range of interactive thresholds. Table III shows how long it takes for the Flask endpoint to respond to repeated single-gate operations.

TABLE III. API RESPONSE LATENCY (100 TRIALS, 3-QUBIT CIRCUIT)

Metric	Latency (ms)
Minimum	2.1
Mean	4.3
Maximum	9.8
Std. Dev.	1.5

Three.js Bloch sphere rendering kept 60 frames per second on all tested browsers (Chrome 122, Firefox 124, Edge 122), which shows that it worked well for interactive use. The main reason the current implementation limits the number of qubits to three is that the statevector dimension (2n) grows exponentially. If you add two more qubits, the state dimension becomes 32 elements, which is still manageable but not needed for the teleportation demonstration.

E. Comparison with Existing Platforms

TABLE IV. FEATURE COMPARISON WITH EXISTING QUANTUM SIMULATION PLATFORMS

Feature	Qiskit	IBM QE	Quirk	Proposed System
Browser-based	No	Yes	Yes	Yes
No install required	No	Partial	Yes	Yes
Real-time Bloch sphere (3D)	Limited	No	Basic 2D	Full 3D
Structured teleportation demo	Code-only	Partial	Partial	Yes
Account required	No	Yes	No	No
Modular backend architecture	Yes	N/A	No	Yes
Real hardware access	Yes	Yes	No	No

Table IV shows that the proposed system is different from all the others because it combines Quirk's browser accessibility with Qiskit's statevector simulator's computational accuracy and a structured educational teleportation demonstration that none of the other platforms have. It also doesn't require users to create an account or install a local SDK. The main drawback compared to IBM QE is that it doesn't have real quantum hardware connectivity, which is something that will be added in the future.

V. Conclusion and Future Work

This paper introduced an interactive web-based quantum circuit simulator for quantum teleportation education, combining Flask, Qiskit, and Three.js into a unified client-server platform. The system provides real-time three-dimensional Bloch sphere visualization, mathematically precise statevector simulation, and a structured five-stage teleportation tutorial that can be accessed from any modern browser without the need for installation or registration.

Experimental validation verified unit fidelity across all tested teleportation inputs, accurate Bloch vector computation for the complete implemented gate set, and interactive response latencies under 10 ms. When compared to other platforms, the proposed system stands out because it combines the accessibility, depth of visualization, and structured educational focus that are all necessary for teaching quantum computing to beginners.

The study shows that it is possible to make complicated quantum protocols easier to understand by carefully combining modern web technologies with established quantum simulation libraries, without losing mathematical rigor.

Future work will go in four different directions. The first step is to connect with IBM Quantum Experience backends so that circuits can run on real quantum processors. This will let students compare the ideal simulation with the noise from the hardware. Second, using Qiskit Aer's depolarizing and thermal relaxation channels to model noise will add realistic decoherence to the teleportation demonstration. Third, the gate set will be expanded to incorporate parameterized rotation gates (R_x, R_y, R_z) with user-configurable angles, facilitating arbitrary state preparation. Fourth, the qubit limit will be raised by using better sparse statevector representations that work with circuits that are important to Grover's and the Deutsch-Jozsa algorithms. There are also plans for support for collaborative multi-user sessions and the ability to export circuits to Qiskit Python scripts.

References

[1]M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2010.

[2]J. Preskill, "Lecture notes for physics 219: Quantum computation," California Institute of Technology, 1998–2021. [Online]. Available: http://theory.caltech.edu/~preskill/ph229/

[3]C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, "Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels," *Phys. Rev. Lett.*, vol. 70, no. 13, pp. 1895–1899, Mar. 1993.

[4]D. Bouwmeester, J.-W. Pan, K. Mattle, M. Eibl, H. Weinfurter, and A. Zeilinger, "Experimental quantum teleportation," *Nature*, vol. 390, no. 6660, pp. 575–579, Dec. 1997.

[5]S. Devitt, W. Munro, and K. Nemoto, "Quantum error correction for beginners," *Rep. Prog. Phys.*, vol. 76, no. 7, p. 076001, 2013.

[6]A. Cross, L. Bishop, J. Smolin, and J. Gambetta, "Open quantum assembly language," arXiv:1707.03429, 2017.

[7]F. Bloch, "Nuclear induction," *Phys. Rev.*, vol. 70, pp. 460–474, Oct. 1946.

[8]IBM Quantum, "Qiskit: An open-source framework for quantum computing," 2023. [Online]. Available: https://qiskit.org/documentation/

[9]Google Quantum AI, "Cirq: A Python library for writing, manipulating, and optimizing quantum circuits," 2023. [Online]. Available: https://quantumai.google/cirq

[10]K. Svore, M. Geller, M. Troyer, J. Azariah, C. Granade, B. Heim, V. Kliuchnikov, M. Mykhailova, A. Paz, and M. Roetteler, "Q#: Enabling scalable quantum computing and development with a high-level DSL," in *Proc. Real World Domain Specific Languages Workshop*, 2018, pp. 1–10.

[11]C. Gidney, "Quirk: A drag-and-drop quantum circuit simulator," 2016. [Online]. Available: https://algassert.com/quirk

[12]IBM Quantum, "IBM Quantum Experience," 2023. [Online]. Available: https://quantum-computing.ibm.com/

[13]E. Weisz and M. Plenio, "Pedagogical effects of Bloch sphere visualization in introductory quantum courses," *Am. J. Phys.*, vol. 89, no. 4, pp. 345–352, 2021.

[14]R. Cabello, "Three.js: JavaScript 3D library," 2023. [Online]. Available: https://threejs.org/docs/

[15]A. Ronacher, "Flask: A micro web framework written in Python," Pallets Project, 2023. [Online]. Available: https://flask.palletsprojects.com/

[16]C. R. Harris et al., "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, 2020.

[17]A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, "Validating quantum computers using randomized model circuits," *Phys. Rev. A*, vol. 100, p. 032328, 2019.

[18]P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Ann. Symp. Foundations Computer Science*, 1994, pp. 124–134.