First Round Review

Bloch Sphere Simulator

Specification

Quantum Computing and its Applications

March 20th, 2025

Team members:
Akylai Atakanova, QRZ5BZ
Klevis Imeri, T4XGKO

I. Introduction

Existing Bloch Sphere Simulators

The Bloch sphere is a fundamental tool in quantum computing that visualizes the state of a single qubit. A Bloch Sphere simulator allows users to observe qubit transformations under various quantum operations, making it an essential educational research tool. In this section, we will explore existing Bloch Sphere simulators and evaluate their advantages and disadvantages. By understanding these tools, we aim to understand their capabilities, limitations, and suitability for different quantum computing applications.

1. **IBM Quantum Platform (Qiskit)**
   Description: IBM Quantum Platform is an online platform that allows public and premium access to IBM's cloud-based quantum computing services. IBM's Qiskit is a popular computing framework that includes tools for visualizing quantum states on the Bloch Sphere. It is widely used for educational and research purposes.
   Advantages:
   - The whole code is on Git Hub (open source).
   - Integration with real quantum hardware and simulators.
   - Extensive documentation and community support.
   - Supports visualization of single-qubit states and gates.
   - For multi-qubit states, Qiskit provides the QSphere.

   Disadvantages:
   - Requires familiarity with Python and Qiskit.
   - It is static. You have to rerun the Python script every time to generate the sphere.
   - The sphere is just an image and not a 3d rendered object.
   - IBM Quantum account requirement and overwhelming interface for beginners.

2. **Quirk**
   Description: Quirk is a drag-and-drop, web-based quantum circuit simulator that includes a Bloch sphere visualization tool. It is great for manipulating and exploring small quantum circuits. Quirk's visual style is intuitive, state displays updates in real-time as you change the circuit, and the general experience is fast and interactive.
   Advantages:
   - The whole code is on Git Hub (open source).
   - Easy to use and accessible through a browser with no setup needed.
   - Real-time visualization of quantum states and gates.
   - Supports multi-qubit simulations.
   - The source code is available for modifications and enhancements.

   Disadvantages:
   - Limited customization options.
   - Does not support multi-qubit Bloch sphere representations.
   - Not suitable for complex quantum computations or advanced simulations.
   - The Bloch sphere visualization is not the tool's primary focus.
   - No integration with real quantum hardware.

3. **Quantum Computing Playground**
   Description: Quantum Computing Playground is a browser-based WebGL Chrome Experiment. It features a GPU-accelerated quantum computer with a simple IDE interface, a scripting language with debugging and 3D quantum state visualization features, and a graphical user interface (GUI).
   Advantages:
   - Easy to use and accessible through a browser, no installation is required.
   - User-friendly interface for beginners.
   - Interactive visualization of quantum states and gates.

   Disadvantages:
   - Limited functionality compared to more advanced tools.
   - Limited documentation and community support.
   - Does not support integration with real quantum hardware.

4. **QuTiP (Quantum Toolbox in Python)**
   Description: QuTiP is an open-source framework written in the Python programming language, designed for simulating the open quantum dynamics of systems. The Quantum computing library includes a Bloch sphere visualization module and is widely used in academia for simulating quantum systems.
   Advantages:
   - Highly customizable and flexible
   - The source code is available for modifications and enhancements.
   - Supports advanced quantum mechanics simulations, optimized for numerical simulators, and leverages multiprocessing to speed up computations.
   - Includes tools for visualizing time evolution on the Bloch Sphere.

   Disadvantages:
   - A steeper learning curve for beginners requires programming knowledge, specifically, users must be comfortable with Python and numerical computing libraries like NumPy and SciPy.
   - Requires installation and setup of Python dependencies.
   - The Bloch Sphere visualization is not as interactive as other tools.
   - No direct quantum hardware integration.
   - Computationally expensive for large systems, and is not designed for large quantum circuits.

5. **Kherb.io**
   Description: A web-based bloch sphere by Konstantin Herb in 2023.
   Advantages:
   - The whole code is on GitHub (open source).
   - The code is simple to understand.
   - Supports visualization of single-qubit states and gates.
   - It supports elementary gates (Pauli, Hadamand, Phase) and more.
   - You can customize it in the settings.

   Disadvantages:
   - No integration with real quantum hardware.
   - No animations.
   - The sphere reverts to its original position every time you apply a transformation.

6. **QuVis**

Description: A web-based Bloch sphere simulator that describes the inner workings of how the positions on the Bloch sphere of a quantum bit are calculated. Its purpose is more educative. Uses the decomposition formula of a bit into the polar and azimuth angles.

Advantages:

- Educative. Step-by-step explanations.
- You can use the angles to place the qubit.

Disadvantages:

- No quantum gates.

There are numerous Bloch sphere simulators available on the internet, each offering unique features and functionalities. Some noteworthy ones include:

- Bloch-Sphere.app – This simulator stands out due to its undo and redo history, which allows users to experiment with different quantum state manipulations efficiently. It also features a modern and intuitive design, resembling the Kherb.io Bloch sphere simulator in its interface and usability.
- Attila Kun's Bloch Sphere: This tool provides an interactive, draggable qubit, which is particularly useful for visualizing state changes in real-time.
- Desmos Bloch Sphere – An implementation of the Bloch sphere simulator entirely within the Desmos 2D graphing calculator.

II. Project Specification

This project is a fronted-only web-based application designed to provide an interactive Bloch sphere simulator. The application consists of two main panels:

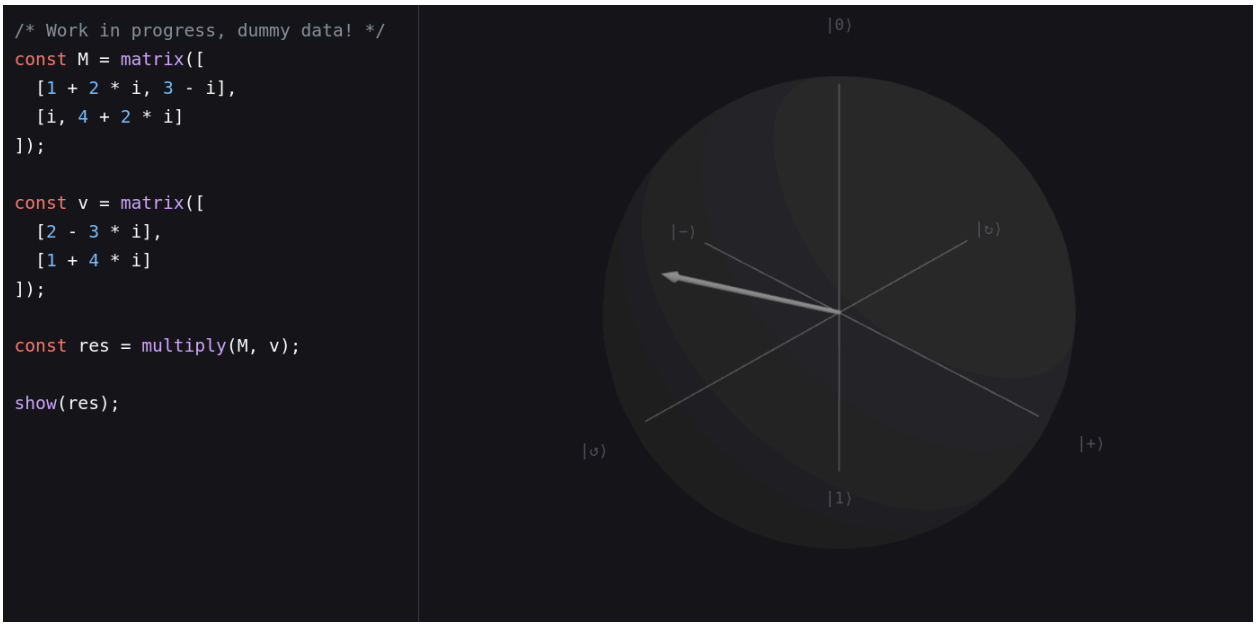
- Bloch Sphere Panel – Displays the Bloch sphere, allowing users to visualize quantum states and transformations dynamically.
- Code Panel – Provides an interface for writing and executing quantum-related code, featuring syntax highlighting for better readability.

System Design and Architecture

The design of these panels is illustrated in the image below. For typography, we have chosen a monospace font, ensuring a clean and readable coding experience. Additionally, we have implemented the GitHub Dark theme for syntax highlighting.

The Bloch sphere is rotatable in the x-y plane using mouse controls or touch gestures on compatible devices. Currently, the program is in a semi-functional state, with ongoing improvements. The program in its current state can be found here. The code can be found here

img 1: UI Interface Demo



Note: For the moment we only support Chrome browsers in desktop environments.

Programming Interface and Functionality

The Bloch sphere simulator provides a flexible scripting interface that allows users to define and manipulate quantum states using JavaScript. It utilizes external libraries such as math.js (for matrix operations) and three.js (for 3D rendering). Users can define matrices and vectors, apply quantum transformations, and visualize the results on the Bloch sphere.

Defining Matrices and Vectors:

In this simulator, matrices and vectors are represented as arrays. A matrix is a two-dimensional array, while a vector is a column matrix. The syntax for defining them is as follows:

```
const M = matrix([
      [a,  b],
      [c,  d]
]);
```

A vector, being a column matrix, is defined as:

```
const v = matrix([
      [a],
      [b]
]);
```

This structure allows users to define qubit states and quantum gates straightforwardly.

Matrix-Vector Multiplication:

Once a matrix and vector are defined, users can apply transformations using matrix multiplication:

```
const res = multiply(M, v);  //Mv = res
```

Chaining multiple operations is also supported:

```
const res = multiply(M, multiply(M, v));  //MMv = res
```

This enables the application of successive quantum operations to a qubit state.

Displaying Vectors on the Bloch Sphere:

The simulator provides built-in functions for visualizing quantum states. To print a vector in the browser console, users can call:

```
console.log(vector);
```

To display the vector on the Bloch sphere, the function users have to use:

```
show(vector);
```

Additionally, multiple vectors can be displayed sequentially, connected by arcs:

```
show(vec1, vec2, vec3, …);
```

Automatic Normalization and Unitary Check:

To ensure correctness, the simulator automatically normalizes vectors before displaying them. Moreover, matrices are checked for unitarity before they can be applied to a quantum state. This ensures that all operations adhere to the fundamental properties of quantum mechanics.

Built-in Quantum Gates:

The simulator provides several predefined quantum gates, including the Pauli gates(X, Y, Z), the Hadamard gate (H), and the Phase gate (P). These gates are available by default in the `script.js` file.
For example, the Hadamard gate is defined as follows:

```
const H = matrix([
  [1/sqrt(2), 1/sqrt(2)],
  [1/sqrt(2), -1/sqrt(2)]
]);
```

Users can apply these gates to transform qubit states and visualize the results dynamically.

Extensibility and JavaScript Capabilities:

The simulator allows users to execute any JavaScript code using the integrated scripting environment. Since the script is simply injected JavaScript in the frontend, users can leverage the full power of JavaScript, including:

- Performing custom mathematical operations with math.js.
- Creating complex visual effects with three.js.
- Sending fetch requests to external servers for additional data processing.

This flexibility makes the simulator not only a powerful visualization tool but also an interactive quantum computing playground for experimentation.

Development Environment and Tools

Selected Tools and Technologies:

- **Svelte** [https://svelte.dev/]: Svelte is a JavaScript web framework that provides a **reactive UI system** and organizes code into structured file and folder paths. Its lightweight and efficient architecture simplifies the management of larger projects.
- **Three.js** [https://threejs.org/]: Three.js is a JavaScript 3D graphics library built on top of WebGL. It enables the rendering of the Bloch sphere, axes, and 3D vectors within a browser canvas.
- **Threlte** [https://threlte.xyz/]: Threlte is a library that integrates Three.js with Svelte, making it easier to create and manage 3D scenes within the Svelte framework.
- **Math.js** [https://mathjs.org/]: Math.js is an extensive mathematical library for JavaScript. It provides built-in support for matrix and vector operations, which are essential for simulating quantum states.
  Example:

```
const H = matrix([..]);
```

- **Git** [https://git-scm.com/]: Git will be used as a distributed version control system to track changes and manage collaboration among team members.
- **GitHub** [https://github.com/KlevisImeri/BlochSphere/]: GitHub will serve as the central repository for our project, facilitating code sharing and version control.
- **Google Docs** [https://docs.google.com/]: Google Docs will be used for writing and maintaining project documentation, ensuring easy collaboration and version tracking.