Halfway! Week#6-7

Rushiraj Gadhvi

2024-10-07

This week marks the halfway point of GSOC 2024, and the project has successfully met its midway goal of completing the development of the Matplotlib Renderer: MatRenderer. Moving forward, my primary focus will be on the Text Based Renderer, which is the major goal for the second half of the project, along with some additional functionalities.

It has been an amazing experience contributing to GSOC'24 at QuTiP so far, and I am excited to continue making progress.

## What did I do this week?

1. **Moving LaTeX plotting**

* Most of my effort week-6 went into understanding and moving the LaTeX code to its separate file from circuit.py to texrenderer.py. A separate class, TeXRenderer, was also created to further organize the code.

1. **Updating the Wire Rendering of MatRenderer**

* In Week-6, I also updated the extende\_wire() function for wire rendering. This change eliminated overlaping wire renderings and calls for extension of wires after each layer rendering. Now, the entire wire is only drawn at the end, after all the gates are rendered.

1. **Built a proof of Concept for Text Rendering**

* As previously mentioned, using functions from the MatRenderer class for the text renderer is the best approach for future code maintenance. During week-7, I developed a basic prototype to show how this concept works. At the moment, this prototype is implemented for the Single Qubit Gate. It effectively reuses the layer management functions in MatRenderer, specifically the manage\_layers() and x\_skip() functions.

#### **Circuit Representation Example**

{  
 top: [render for the top of the 1st wire, 2nd wire, 3rd wire ...],  
 mid: [render for the middle of the 1st wire, 2nd wire, 3rd wire ...],  
 bot: [render for the bottom of the 1st wire, 2nd wire, 3rd wire ...],  
}

#### **Better Examples**

**Example-1**

qc = QubitCircuit(2)  
qc.add\_gate("H", targets=[0])  
qc.add\_gate("RZ", targets=[1], arg\_value=0.1)

┌───┐   
q0 :───┤ H ├───  
 └───┘   
 ┌────┐   
q2 :───┤ RZ ├──   
 └────┘

*This would be stored as*

{  
 top: [" ┌───┐ ", " ┌────┐ "],  
 mid: ["q0 :───┤ H ├───", "q2 :───┤ RZ ├──"],  
 bot: [" └───┘ ", " └────┘ "],  
}

**Example 2**

qc = QubitCircuit(1)  
qc.add\_gate("H", targets=[0])  
qc.add\_gate("H", targets=[0])

┌───┐ ┌───┐   
q0 :───┤ H ├──┤ H ├─  
 └───┘ └───┘

*The layer increment would look like,*

# STEP 1  
{  
 top: [" "],  
 mid: ["q0 :──"],  
 bot: [" "]  
}   
  
# STEP 2  
{  
 top: [" ┌───┐ "],  
 mid: ["q0 :───┤ H ├─"],  
 bot: [" └───┘ "]  
}   
# STEP 1  
{  
 top: [" ┌───┐ ┌───┐ "],  
 mid: ["q0 :───┤ H ├──┤ H ├─"],  
 bot: [" └───┘ └───┘ "]  
}

## Plan for Next Week

* Develop the base class for both MatRenderer and TextRenderer.
* Implement tests for MatRenderer.
* Add alignment options for gate layers to ensure gates in a layer are centered relative to each other.
* Investigate and implement multi-qubit gate support in the text renderer.
* Finalize moving code from latex\_circuit.py to texrenderer.py.
* Organize fixed constants in MatRenderer using a style dictionary or data class.