Quantum Computing Project

Release 0.01

Conner Adlington, Julia Bauer, etc.

Mar 11, 2024

CONTENTS:

CHAPTER

ONE

NAME THE CHAPTER UP HERE

This narrative can explain the whole point of the module (the file).

You can include maths inline like this: $a^2 + b^2 = c^2$

Or, if you need to display, do it like this (be sure to leave a blank line above the .. math:: delcaration):

$$\lim_{n \to \infty} \frac{1}{n} \neq \infty$$

We can make use of referencing code objects like classes and methods like this:

In this chapter we will start by explaining the *Example* class and the required arguments for the `__init__()` method.

Note: Add notes like this.

Warning: Add warnings like this.

Insert code sample with backticks *def __init__()*. Make text **bold**, or *italics* just like markdown.

This is a list:

- With
- Bullets.

This is also a list:

1. With
2. Numbers.

class example.**Example**(*arg1: int*)

Class explanation.

This can be a high level overview of what the object is.

arg1

Describe the *self.arg1* term. This behaviour is unique to the init method.

usefulMethod(*input: int*)

Explain the utility of this method.

Params:

input:
This `int` should represent something.

Returns:
This method returns a `Vector`.

`class` example.**Example2**

This is the second class

CHAPTER

TWO

GATES MODULE

Note: This code is still under development and until the version is incremented to a 1.* you should not trust any of this documentation.

class gates.**Gate**(*dimension: int*)

This is the gate class which contains the gates we need to implement Grover's algorithm.

When we initialise, we declare the dimension of the quantum register so that the gates (except the Hadamard) can be provided in the correct dimension.

The Hadamard gate is scaled (using tensor products) at the point of implementation to make the application more obvious.

h()

Single-qubit Hadamard gate.

This gate is a π rotation about the X+Z axis, and has the effect of changing computation basis from $|0\rangle, |1\rangle$ to $|+\rangle, |-\rangle$ and vice-versa.

Matrix Representation:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

CHAPTER

THREE

TENSOR MODULE TEST SUITE

This module tests the classes included in the tensor module.

class tests.test_tensor.**TestOperator**(*methodName='runTest'*)

test_operator_tensor_product_vs_notes()

Tensor Product of Hadamard, Identity and Hadamard, as presented in the slides - $H \otimes \mathbb{I} \otimes H$.

This manually creates the hadamard gates and performs the tensor product using the `tensor()` method.

The result is compared against the result in the slides:

$$H \otimes \mathbb{I} \otimes H = \frac{1}{2} \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 & -1 & -1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & -1 & -1 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 \end{pmatrix}$$

class tests.test_tensor.**TestVector**(*methodName='runTest'*)

This class aims to provide end to end testing of the Vector class within the tensor module.

The purpose of each test and the method by which it is validated is outlined below.

test_vector_construction()

This test aims to check…

PYTHON MODULE INDEX

INDEX