|  |
| --- |
|  |
| qvantum – python module |
| Documentation |

|  |
| --- |
| Sohonyai, Ádám  2018.11.18. |

Contents

[1 Introduction 2](#_Toc48435)

[2 Installing 2](#_Toc48436)

[2.1 pip install 2](#_Toc48437)

[2.2 wheel install 2](#_Toc48438)

[2.3 setup file 2](#_Toc48439)

[3 Module Classes 2](#_Toc48440)

[3.1 qubit class 3](#_Toc48441)

[3.2 register class 3](#_Toc48442)

[3.3 gate class 3](#_Toc48443)

[3.4 layer class 3](#_Toc48444)

[3.5 circuit class 3](#_Toc48445)

[3.6 bloch class 3](#_Toc48446)

[4 Examples 3](#_Toc48447)

[4.1 Quantum teleportation 3](#_Toc48448)

[4.2 Grover’s algorithm 3](#_Toc48449)

[5 Notes 3](#_Toc48450)

[5.1 Module reading error 3](#_Toc48451)

[5.2 Deleting a qubit from register 3](#_Toc48452)

[5.3 Ѱ sign in python2 3](#_Toc48453)

# Introduction

qvantum is a python module with the goal to ensure an easy use library for understanding quantum computing better or designing new quantum algorithms. Working with this module helps you the get more familiar with the basic concepts such as qubit, register or quantum gate, meanwhile the tool has the power for deeper analysis and development.

The module is in beta release phase: tested but it might contain bugs, therefore every constructive note is highly welcomed. Also If you would like to collaborate in the developing process then do not hesitate and contact us.

# Installing

qvantum module can be easily installed using three different approach below.

## pip install

The latest version of the module can be installed online from the PyPi page using pip in command line:

pip install qvantum

or

python –m pip install qvantum

or

python -m pip install --index-url <https://test.pypi.org/simple/> qvantum

## wheel install

The latest version of the module can be downloaded from the PyPi page in .whl format which can be used for installation:

pip install qvantum-x.xx-py2.py3-none-any.whl

or

python –m pip install qvantum-x.xx-py2.py3-none-any.whl

## setup file

A setup.py file is also available on PyPi page. Download the file then run the command in the folder where the setup.py file was downloaded. Use –e if you want the module be immediately available for every user in your system:

pip install . (or pip install –e .)

or

python –m pip install . (python –m pip install –e .)

# Module Classes

In qvantum module there are some classes which represents the basic objects in quantum computing such as: a qubit, a register, a gate, a layer and a circuit. These objects (and therefore the classes which represents them) are built on each other. Due to this concept a register is built on qubits, layers are formed by gates and circuits are created out of gates.

There is a sixth class, the bloch class which is used for teh Bloch representation and visualisation of a qubit.

## Qubit class

In quantum computing a qubit or quantum bit is the basic unit of quantum information. Every qubit has two clear states such as 0 and 1 but unlike a classical bit a qubit can be in the mixture of these clear states.

An instance of the qubit class has these methodes below.

qvantum.Qubit.\_\_init\_\_(alpha, beta)

Method to initialize an instance of the qubit class. The squared sum of alpha and beta bust be equal to zero otherwise a ValueError will be thrown.

Parameters:

alpha: int, float or complex

beta: int, float or complex

Examples:

import math

import qvantum

q = qvantum.Qubit(1, 0)

qvantum.Qubit(1 / math.sqrt(2), 1 / math.sqrt(2))

q = qvantum.Qubit(5, 2)

q = qvantum.Qubit(1, ’red’)

qvantum.Qubit(1, 0).show()

qvantum.Qubit.get\_alpha()

Getter method of alpha.

Examples:

import qvantum

q = qvantum.Qubit(1, 0)

q.get\_apha()

qvantum.Qubit.get\_beta()

Getter method of beta.

Examples:

import qvantum

q = qvantum.Qubit(1, 0)

q.get\_beta()

qvantum.Qubit.set\_amplitudes(alpha, beta)

Setter method to replace the old coefficients to new ones. The squared sum of alpha and beta bust be equal to zero otherwise a ValueError will be thrown.

Parameters:

alpha: int, float or complex

beta: int, float or complex

Examples:

import math

import qvantum

q = qvantum.Qubit(1, 0)

q.show()

q.set\_amplitudes(0, 1)

q.show()

qvantum.Qubit.show()

This method shows the state of the qubit object.

Examples:

import qvantum

q = qvantum.Qubit(1, 0)

q.show()

qvantum.Qubit.measure()

This method performs a measurement on the qubit and returns with one clear state by the distribtion according to the coefficients.

Examples:

import qvantum

q = qvantum.Qubit(x, y)

q.show()

q.measure()

q.show()

qvantum.Qubit.ket()

This method returns with the ket vector representation of the qubit.

Examples:

import qvantum

q = qvantum.Qubit(x, y)

q.ket()

qvantum.Qubit.bra()

This method returns with the bra vector representation of the qubit.

Examples:

import qvantum

q = qvantum.Qubit(x, y)

q.bra()

qvantum.Random\_Qubit.\_\_init\_\_()

This is an inhereted class from the Qubit class. They share the same methods but when an instance of the Random\_Qubit class is created the coefficients are randomly choosen.

Examples:

import qvantum

rq = qvantum.Random\_Qubit()

rq.show()

qvantum.Random\_Qubit().show()

qvantum.check\_qubit.py

This file contains the decorators and is used to check the arguments when a method is called with parameters.

## register class

xxx

## gate class

xxx

## layer class

xxx

## circuit class

xxx

## bloch class

xxx

# Examples

xxx

## Quantum teleportation

xxx

## Grover’s algorithm

xxx

# Notes

xxx

## Module reading error

xxx

## Deleting a qubit from register

xxx

## Ѱ sign in python2

xxx