#16 Julia in Qiskit, QuantumCircuits library.

Rafał Pracht

Mentored by John Lapeyre and Jim Garrison

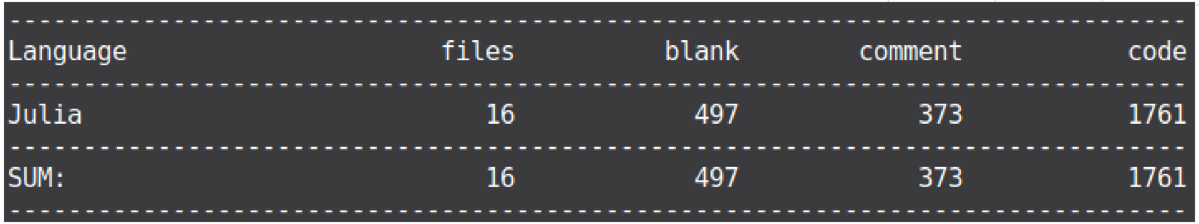
Description

The idea of the project was to the improvement of qiskit-alt, a high-performance Julia backend for Qiskit with a Python frontend. During the discussion, John and Jim agreed to mentor me on my work on open-source Julia library QuantumCircuits which use Qiskit to execute the circuit on real devices.

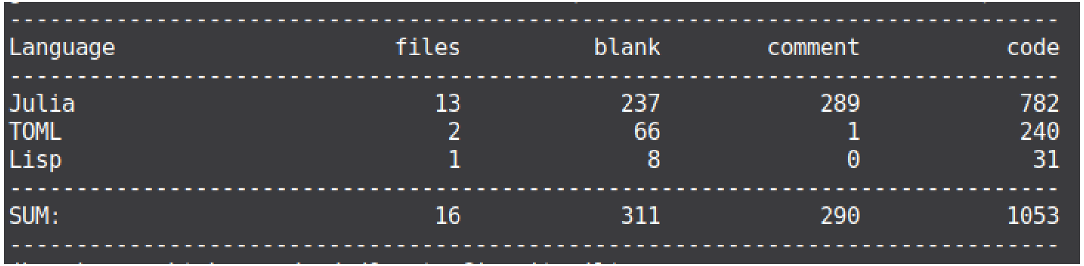
QuantumCircuits

open-source - Apache License 2.0

https://github.com/Adgnitio/QuantumCircuits.jl



```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Julia                           16            497            373           1761
-------------------------------------------------------------------------------
SUM:                            16            497            373           1761
-------------------------------------------------------------------------------
```



```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Julia                           13            237            289            782
TOML                             2             66              1            240
Lisp                             1              8              0             31
-------------------------------------------------------------------------------
SUM:                            16            311            290           1053
-------------------------------------------------------------------------------
```

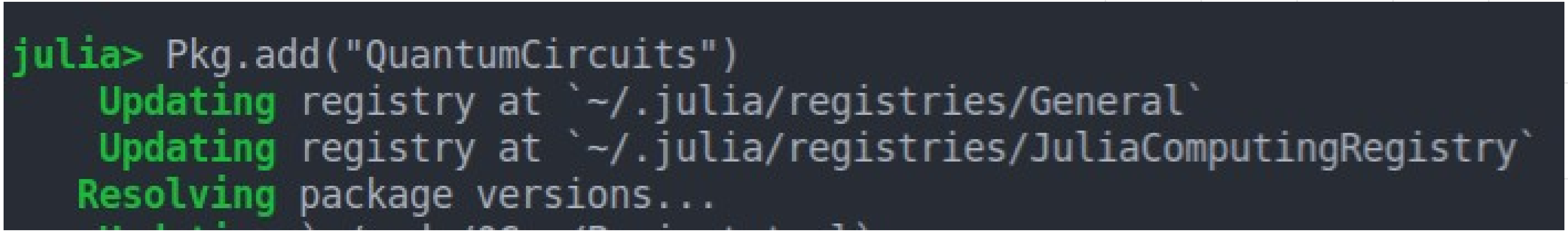
Why?

- I love Julia, and Quantum :)

- Possibility of learning

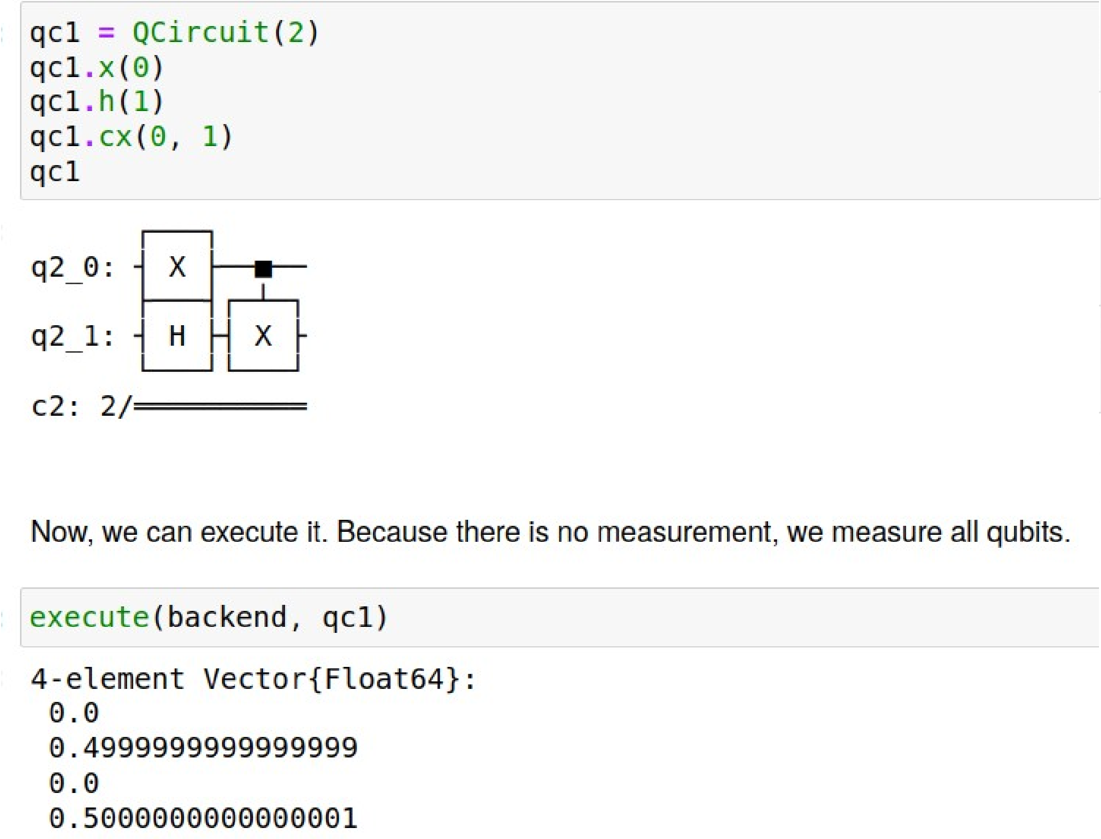
- Problems, that I encountered when using QML in Qiskit.

Installation



```
julia> Pkg.add("QuantumCircuits")
    Updating registry at `~/.julia/registries/General`
    Updating registry at `~/.julia/registries/JuliaComputingRegistry`
   Resolving package versions...
```

Use



```
qc1 = QCircuit(2)
qc1.x(0)
qc1.h(1)
qc1.cx(0, 1)
qc1
```

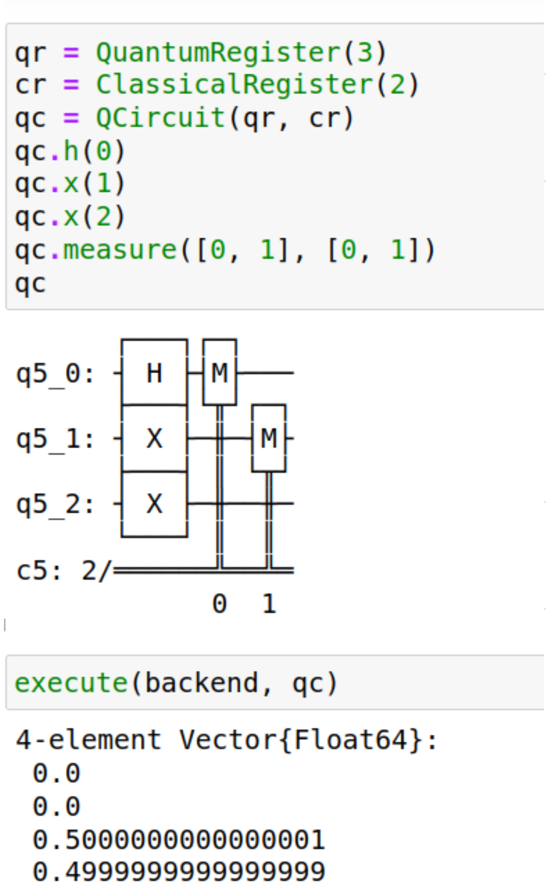
```
      ┌───┐     
q2_0: ┤ X ├──■──
      ├───┤┌─┴─┐
q2_1: ┤ H ├┤ X ├
      └───┘└───┘
c2: 2/══════════
```

Now, we can execute it. Because there is no measurement, we measure all qubits.

```
execute(backend, qc1)
```

```
4-element Vector{Float64}:
 0.0
 0.4999999999999999
 0.0
 0.5000000000000001
```

Use



```
qr = QuantumRegister(3)
cr = ClassicalRegister(2)
qc = QCircuit(qr, cr)
qc.h(0)
qc.x(1)
qc.x(2)
qc.measure([0, 1], [0, 1])
qc
```

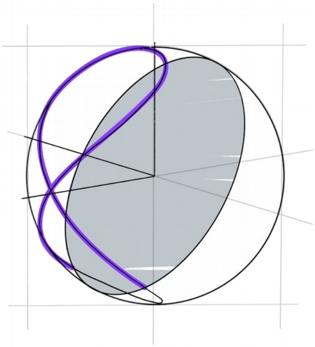
```
       ┌───┐┌─┐   
q5_0: ┤ H ├┤M├───
       ├───┤└╥┘┌─┐
q5_1: ┤ X ├─╫─┤M├
       ├───┤ ║ └╥┘
q5_2: ┤ X ├─╫──╫─
       └───┘ ║  ║ 
c5: 2/══════╩══╩═
             0  1 
```

```
execute(backend, qc)
```

```
4-element Vector{Float64}:
 0.0
 0.0
 0.5000000000000001
 0.4999999999999999
```

QML Issues 1

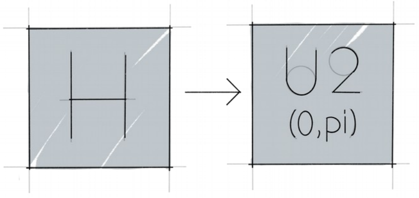
According to my knowledge, Qiskit allows using only a parameter-shift rule to calculate the derivatives. In QuantumCircuits I use the Zygote library to calculate the gradient of the circuits. Thanks to this, the QML algorithms run much faster on the simulator.

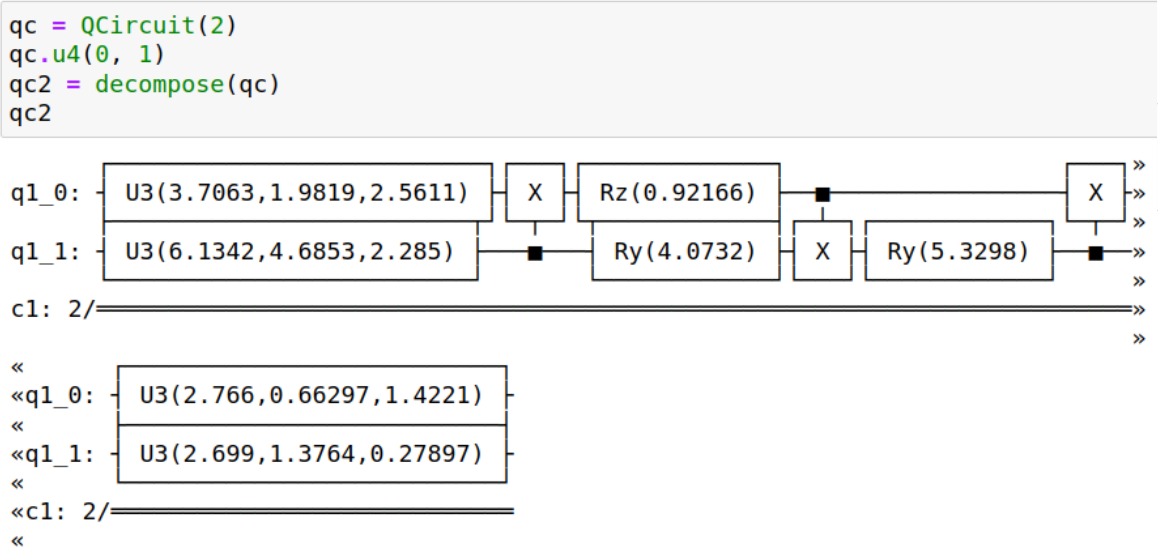


QML Issues 2

In Qiskit, there is no easy method to define loss function comparing two unitaries.

Parameter finding for Cartan's KAK Decomposition decomposition.





```
qc = QCircuit(2)
qc.u4(0, 1)
qc2 = decompose(qc)
qc2
```

```
      ┌──────────────────────────┐┌───┐┌─────────────┐                              ┌───┐»
q1_0: ┤ U3(3.7063,1.9819,2.5611) ├┤ X ├┤ Rz(0.92166) ├──■───────────────────────────┤ X ├»
      ├─────────────────────────┬┘└─┬─┘├─────────────┤┌─┴─┐┌────────────┐           └─┬─┘»
q1_1: ┤ U3(6.1342,4.6853,2.285) ├───■──┤ Ry(4.0732)  ├┤ X ├┤ Ry(5.3298) ├─────────────■──»
      └─────────────────────────┘      └─────────────┘└───┘└────────────┘                »
c1: 2/═══════════════════════════════════════════════════════════════════════════════════»
                                                                                         »
«      ┌──────────────────────────┐
«q1_0: ┤ U3(2.766,0.66297,1.4221) ├
«      ├──────────────────────────┤
«q1_1: ┤ U3(2.699,1.3764,0.27897) ├
«      └──────────────────────────┘
«c1: 2/════════════════════════════
«
```

Future works

The library is in the alpha stage, it works but:

- There is no documentation.

- There are issues in tests for different Julia, Python, and Qiskit versions.

- I would like to do some refactoring and improvement in the library.

- I would like to implement in the library the algorithm for derivative pricing using Quantum Monte Carlo.