Design Your Own Quantum Simulator with R

Indranil Ghosh

Jadavpur University

indranilg49@gmail.com

19/06/2020

Quantum Computation

Qubits and qutrits are the basic units of quantum computation. A qubit is a two label system and a qutrit is a three label system. A quantum environment is developed at first in the R ecosystem:

```
Qu <- new.env(parent=emptyenv())
```

1. Qubits: $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ represented by:

```
Qu$Q0 <- matrix(c(1, 0), ncol=1, byrow=TRUE)
Qu$Q1 <- matrix(c(0, 1), ncol=1, byrow=TRUE)
```

2. Qutrits:$|0\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$, $|1\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ and $|2\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ represented similarly by `Qu$Qt0`, `Qu$Qt1`, `Qu$Qt2`
3. To generate a new quantum system, for example $|110\rangle = |1\rangle \otimes |1\rangle \otimes |0\rangle$, we write:

```
Qu$Q110 = kronecker(Qu$Q1, kronecker(Qu$Q1, Qu$Q0))
```

Quantum Logic Gates

1. Pauli-X gate/ qubit flipping operator: $\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, represented in R by:

```
Qu$sigmaX <- matrix(c(0, 1, 1, 0), ncol=2, byrow=TRUE)
```

We know, $|1\rangle = \sigma_x |0\rangle$, and can be verified

```
Qu$sigmaX %*% Qu$Q0 == Qu$Q1
```

2. Hadamard gate: $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, represented in R by:

```
Qu$H <- matrix(c(1, 1, 1, -1), ncol=2, byrow=TRUE)/sqrt(2)
```

3. Other 2X2 matrices representing 2-system logic gates and Gell Mann matrices representing 3-system logic gates can be generated in the similar fashion.

A simple quantum Algorithm

A simple quantum Algorithm represented by the circuit model starts by initializing a quantum state and then performing unitary gate operations on the circuit before the final measurement. A simple circuit:

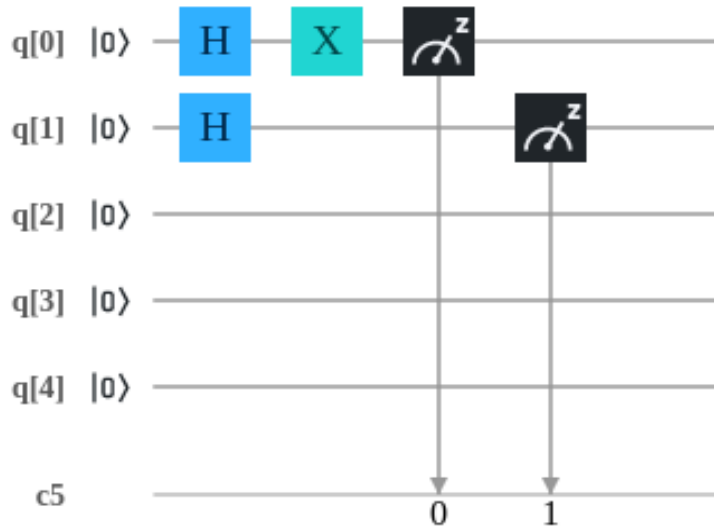


Figure: Circuit (Generated with IBM Q Experience)

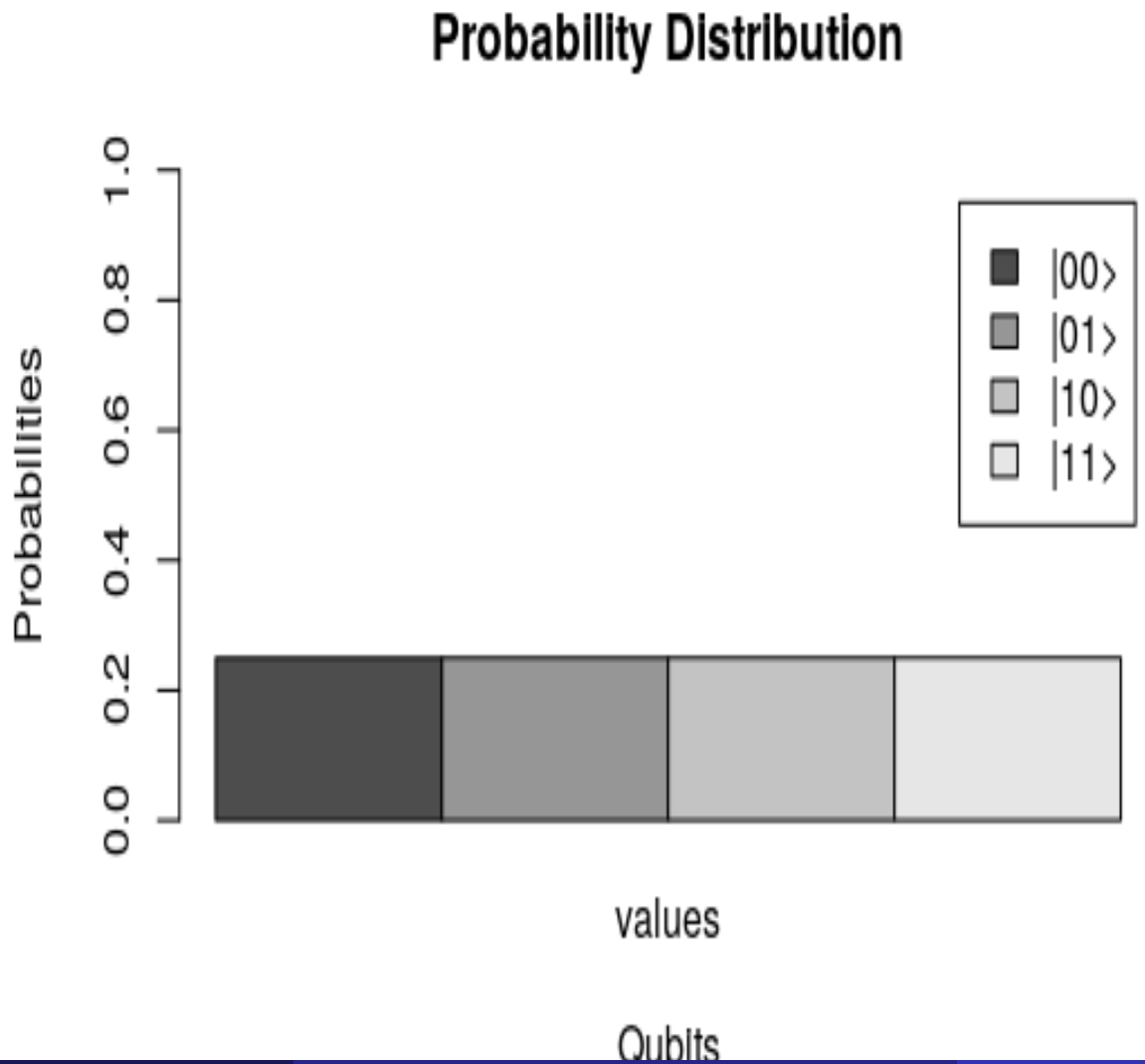
The simulation code in R:

A simple quantum Algorithm

```
Psi <- kronecker(Qu$Q0, Qu$Q0)
HH <- kronecker(Qu$H, Qu$H)
Qu$I <- matrix(c(1, 0, 0, 1), ncol=2, byrow=TRUE)
XI <- kronecker(Qu$sigmaX, Qu$I)
Psi1 <- HH %*% Psi
Psif <- XI %*% Psi1
values <- c()
for (i in 1:length(Psif)){
  values <- c(values, abs(Psif[i])**2)
}
p <- t(as.data.frame(values))
colnames(p) <- c("|00>", "|01>", "|10>", "|11>")
barplot(t(as.matrix(p)), beside=TRUE, legend.text = colnames(p),
        xlab='Qubits', ylab='Probabilities', ylim=0:1,
        main='Probability Distribution')
```

A simple quantum Algorithm

The porbability distribution plot after measurement:



Quantum Game Theory

- My recent work has been developing an R package, *QGameTheory* that helps in simulating quantum versions of some Game Theoretic Models. This package can also be used to perform simple quantum computing simulations in the ways mentioned above.
- CRAN link: https://cran.r-project.org/web/packages/QGameTheory/index.html
- Implementing quantum moves by replacing the classical ones in game theory, it can be seen that interesting new results may appear. For example, in Quantum Prisoner's Dilemma, one can actually escape the dilemma that rises in the otherwise classical case.

References

Indranil Ghosh (2020)

QGameTheory

CRAN https://cran.r-project.org/web/packages/QGameTheory/index.html

Michael Nielsen and Isaac Chuang (2010)

Quantum Computation and Quantum Information

ISBN:978-1-107-00217-3.

J. Orlin Grabbe (2005)

An Introduction to Quantum Game Theory

arXiv:quant-ph/0506219

The End