Synthesis of arbitrary quantum gates with IBMQ

Debarghya Kundu, *IIT Guwahati*

***Abstract*—This article focuses on the implementation of any arbitrary unitary matrix i.e any quantum gate using the library set of gates available in IBM Quantum Experience. A few optimization technique has been discussed with their rigorous implementation in python. I would like to express my sincere gratitude to Prof. Subhamoy Maitra for constant support and motivation through the progress of the project during my summer internship, 2018 at ISI Calcutta.**

I. Introduction

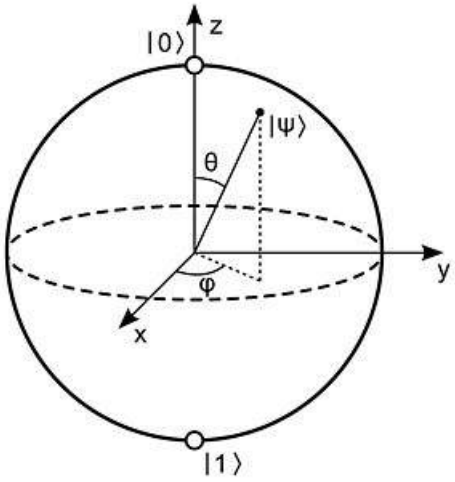


Fig. 1. Bloch Sphere Representation

A qubit is a two-level quantum mechanical system much like a classical bit(0 or 1), however which has additional weird properties like superposition i.e it can be in both the basis states $|0\rangle \; and \; |1\rangle$ simultaneously. Qubits form the fundamental unit of quantum computations. It is often represented by the symbol ψ where any qubit may be written as $\psi = \alpha\,|0\rangle + \beta\,|1\rangle$ where $\alpha, \beta \in \mathbb{C}$. The possible quantum states for a single qubit can be visualized on a bloch sphere(fig-1).

$$\psi = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle \tag{1}$$

Quantum logic gates are the basic building blocks in a quantum circuit analogous to our conventional boolean logic gates in electronic circuits. Unlike boolean gates quantum gates are reversible. Each quantum gate is represented by a unitary matrix which also ensures their reversibility i.e if $2^n x 2^n$ matrix U represents a quantum gate then $UU^\dagger = I$.

Here, we plan to implement the gray-code construction [1] for decomposing any arbitrary n qubit quantum gate into 2x2 unitary matrices and CNOT gates and also probe into certain optimization strategies for fault-tolerant construction of gates. We will be using the ibmq quantum computer interfaced online through python-qiskit module.

II. Theory and Implementation

We adopt a top-down approach for synthesizing a n-qubit unitary into CNOTS and 1-Q gates. We start by factoring the matrix(say U) into two-level matrices[1].

$$U = V_1 V_2 V_3 ..... V_k, \qquad k < \frac{d(d-1)}{2} \tag{2}$$

Algorithm 1 Recursive routine for Two-level Decomposition

function TLD(Gate U,dimension d,level l,Vset)
  if $l == 1$ **then**
    $Vk \leftarrow I_d$
    $Vk_{d-1,d-1} \leftarrow U^*_{d-1,d-1}$
    Vset.append(Vk)
    return Vset
  $i \leftarrow 1 \quad pad \leftarrow d - l$
  while $i < l$ **do**
    if $U_{i+pad,pad} = 0$ **then**
      $i \leftarrow i + 1$
      continue
    $V \leftarrow I_d$
    $a \leftarrow U_{pad,pad}$
    $b \leftarrow U_{i+pad,pad}$
    $V_{pad,pad} \leftarrow a^*/norm(a,b)$
    $V_{i+pad,0+pad} \leftarrow b/norm(a,b)$
    $V_{pad,i+pad} \leftarrow b^*/norm(a,b)$
    $V_{i+pad,i+pad} \leftarrow -a/norm(a,b)$
    $U \leftarrow VX$
    $Vset.append(V)$
    $i \leftarrow i + 1$
  return TLD(U,d,l-1,Vset)

The two-level matrices have a special operational property which is exploited in the following sections. Any $2^n x 2^n$ two-level matrix V when multiplied with the vectors from the set of n qubit basis states gives only 2 non-trivial solutions.

$$\begin{bmatrix} 1 & 0 & 0 & ..... & 0 \\ 0 & \mathbf{a_{ii}} & 0 & ...\mathbf{a_{ij}}.. & 0 \\ . & & & & \\ 0 & \mathbf{a_{ji}} & 0 & ..\mathbf{a_{jj}}.. & 0 \\ . & & & & \\ 0 & 0 & 0 & ..... & 1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ . \\ \omega_k \\ . \\ \omega_{2^n} \end{bmatrix} = \begin{bmatrix} \omega_1 \\ a_{ii}\omega_i + a_{ij}\omega_j \\ . \\ a_{ji}\omega_i + a_{jj}\omega_j \\ . \\ \omega_{2^n} \end{bmatrix} \tag{3}$$

Now, for a n qubit basis state only for one k,$\omega_k = 1$ and $\omega_j = 0 \; \forall$ j $\in \{1, 2...2^n\} - \{k\}$. So,for non-trivial operation of V either ω_i or ω_j has to be 1. So only two n-qubit states allow non-trivial operations on them. V can be uniquely reduced to a 2x2 matrix as

$$\tilde{V} = \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix} \tag{4}$$

Thus $\tilde{V}$ operating on single qubits through a clever gray-code construction[1] with CNOTS can be used to represent the exact unitary operation by V.

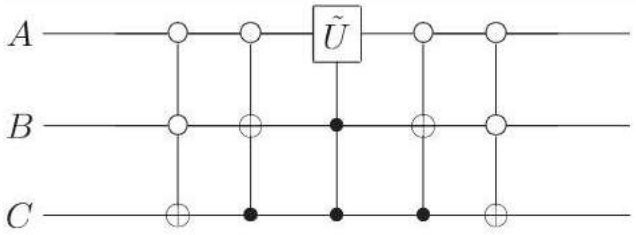


Fig. 2. Circuit implementing the two-level opration.

Any 2^n column vector for n qubit is obtained by kronecker product of n vectors of single qubit. The following routine generates the non-trivial qubits for eg $|A, B, C\rangle$ for a two-level matrix V.

Algorithm 2 Non-Trivial Qubits

Following routine returns the two non-trivial sequence of qubits for matrix V

```
function MOD(m,n)
    if (m%n) = 0 then
        return n
    else
        return m % n
function NTVL(i,j,n)
    g1, gm ←'','' 
    r ← 0
    while ++r ≤ n do
        u ← (n − r − 1)
        if MOD(i, 2^u) ≤ 2^(u−1) then
            g1 ← g1 +' 0'
        else
            g1 ← g1 +' 1'
        if MOD(j, 2^u) ≤ 2^(u−1) then
            gm ← gm +' 0'
        else
            gm ← gm +' 1'
```

Following the gray-code construction and reducing every two-level matrix V_i we can simulate the target matrix U with 2x2 matrices and CNOTs.

Ibmq quantum experience offers only limited set of gates like Pauli matrices,Hadamard,Phase Gate,$\frac{\pi}{8}$-gate and their adjoints. Our goal will be build our circuit upon this gates. The next section offers ways in which we implement some special operators using this gates.

III. Simulating Controlled-U gates

Now that our basic layout of the circuit for U has been given we show ways to construct the n-Q tofolli and n-Q controlled-U gates. In fact, tofollis are controlled-U gates where $U = X$.

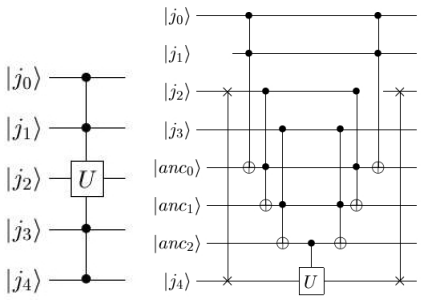


Fig. 3. Equivalent circuits for controlled-U with ancilla-bits

As demonstrated above (in fig-3) the controlled-U operation can be reduced to $3Q$-tofolli gates and Controlled-U operation. We used some ancilla bits in our circuit which are initialized to $|0\rangle$. This usage of extra-qubit as work-bits/ancilla reduces total qubits available for computation (For $n-$qubit gates we require $n-2$ ancilla bits), however since ibmqx5 offers 18-qubits through qiskit we provide a model for synthesizing upto 10 qubit gates. However there are method like bootstrapping a borrowable ancilla bits[2] and using square root gates,which does not require use of this extra qubits but only add to its complexity[2].

Now, that we can conclusively claim that reducing the tofollis and C_U gates,factors the circuit in figure-2 into $1Q$-gates and CNOTS. Our next goal would be show how we reduce tofollis and C_U gates into them. Shown below are construction of Tofolli gates and Swap Gates used extensively in the construction.

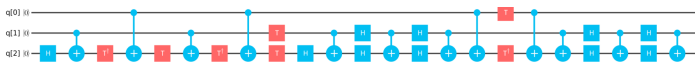


Fig. 4. Tofolli gates

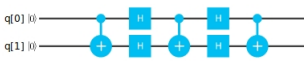


Fig. 5. Swap gates

A. Decomposing U into $e^{i\phi}AXBXC$.

Ibmq does not provide any gate for control operations but CNOTS. So we use a construction[2] to reduce U into

$1Q-$gates and CNOTS. Following figure depicts the construction.

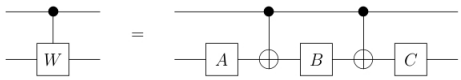


Fig. 6. Controlled-U gates

For any special unitary matrix W,there exists matrices A, B and $C \in SU(2)$ s.t $ABC = I$ and $W = A\sigma_x B\sigma_x C$. A, B, C can be simulated using advanced $U_1(\lambda)$ and $U_3(\theta, \phi, \lambda)$ gates in ibmq. However this does not ensure a fault tolerant construction of gates. To obtain a fault tolerant construction we would use Hadamard and T gates for approximating qubit unitaries to arbitrary accuracy.

IV. Performance of the Construction

Now, that the basic idea of the construction has been laid out we test the output of the program. To test the program import **synth** from the synthesis module. When the following $U = H \otimes X \otimes H \otimes H \otimes X$ and the initial state of $|0, 0, 0, 0, 0\rangle$ is taken as an example the output state of $|q_1, 1, q_2, q_3, 1\rangle$ where q_i's can be 0 or 1 and each with equal probability should be obtained which is consistent with our result.

synth$(U, 32, 5)$

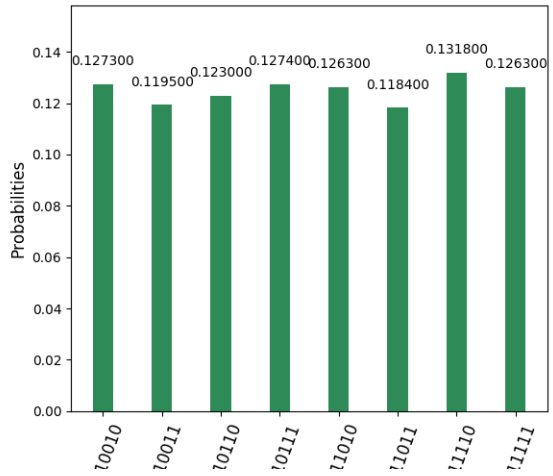


Fig. 7. $U = H \otimes X \otimes H \otimes H \otimes X$

The time taken for any unitary matrix to factor into CNOTS and $1-Q$ gates is also plotted against the no of qubits(in fig-8) on which the gate is applied. Total run-time of the algorithm will highly depend on the system and if run on a real quantum computer(**ibmqx5**) through API call on the priority queue of the live experiments as well.

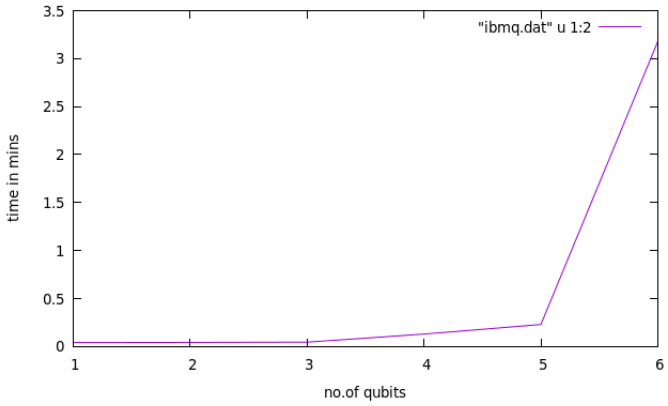


Fig. 8. Runtime of the construction.

Theoretically, the construction requires $\mathcal{O}(n^2 4^n)$ gates from ibmq for n qubit unitaries. However it does not ensure a fault-tolerant result. To obtain a fault tolerant result as we will see in the next section would require large no of gates (H and T gates) and we can also optimize it further with Solovay-Kitaev algorithm.[3]

V. Approximating Unitary Matrices.

Given an unitary matrix it is not always possible to obtain a exact finite sequence of known gates which simulates the matrix. However,Hadamard and T gates can be used to approximate qubit unitaries to arbitrary accuracy. Consider the sequences,

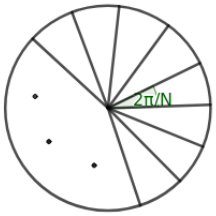
$$THTH = \cos^2\tfrac{\pi}{8} - \iota[\cos\tfrac{\pi}{8}(X+Z) + \sin\tfrac{\pi}{8}Y]$$

$$= R_{\hat{n}}(\theta_0), \text{ where , } \cos(\tfrac{\theta_0}{2}) = \cos^2\tfrac{\pi}{8} \text{ and,}$$

$$\hat{n} = (\cos\tfrac{\pi}{8}\ \sin\tfrac{\pi}{8}, \cos\tfrac{\pi}{8}).$$

We claim that repeated application of $R_{\hat{n}}(\theta_0)$ can generate any $R_{\hat{n}}(\alpha)$ to accuracy ϵ.

Proof:
Let an accuracy of ϵ be required,$\epsilon \in \mathbb{R}^+$
By Archimedean property, $\exists N \in \mathbb{N}$ s.t $N > \frac{2\pi}{\epsilon}$
Define : $\theta_k = k\theta_0 \mod 2\pi$
We claim that $\exists j, k \in \{1, 2, ..., N\}$ s.t $|\theta_j - \theta_k| \leq \epsilon$ (5).



Consider a circle with N sectors marked $S_1, S_2, S_3...$ and angle of each sector $2\pi/N$. Without loss of generality , let $\theta_1 = \theta_0$ lie in sector S_1 where angles are measured

anticlockwise from x axis. If at least, any two θ j,k lie in the same sector then by construction we get $|\theta_j - \theta_k| < \epsilon$. If not then all the θ's must lie in unique sectors, so, $\Sigma_{k=1}^{N-1}|\theta_{k+1} - \theta_k| + |\theta_N - \theta_1| = 2\pi$.If $|\theta_{k+1} - \theta_k| \leq 2\pi/N$ we are done, else assume $|\theta_{k+1} - \theta_k| > 2\pi/N$.So $\Sigma_{k=1}^{N-1}|\theta_{k+1} - \theta_k| + |\theta_N - \theta_1| = 2\pi$ implies $|\theta_N - \theta_1| \leq 0$, we are done!

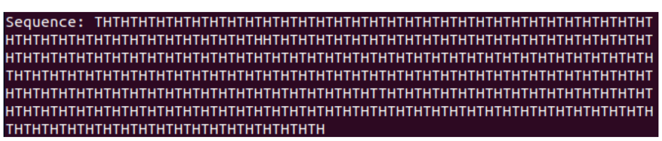
Thus from equation (5) we get $|\theta_j - \theta_k| \leq \epsilon$, or, $|\theta_{k-j}| < \epsilon$. $(k > j)$. Thus given $l \in \mathbb{N}, \theta_{l(k-j)}$ fills up $[0, 2\pi)$ with least count ϵ. So we can get some l_0, such that $E(R_{\hat{n}}(\alpha), R_{\hat{n}}(\theta_0)^{l_0}) < \epsilon$. In a similar way, if we take $\hat{m} = (\cos\frac{\pi}{8} - \sin\frac{\pi}{8}, \cos\frac{\pi}{8})$ then $E(R_{\hat{m}}(\alpha), R_{\hat{m}}(\theta_0)^{l_0}) < \epsilon$, for some l_0, where $R_{\hat{m}}(\theta_0) = HTHT$.

Given any unitary U, it can be written as $R_{\hat{n}}(\theta_1)R_{\hat{m}}(\theta_2)R_{\hat{n}}(\theta_3)$ as $\hat{m}, \hat{n}$ are non-parallel vectors. We can find integers n_1, n_2, n_3 s.t $E(U, R_{\hat{n}}(\theta_1)^{n_1}R_{\hat{m}}(\theta_2)^{n_2}R_{\hat{n}}(\theta_3)^{n_3}) < \epsilon$.

On implementing the idea, we list down the average no of gates required for approximating a random unitary gate against a arbitrary accuracy ϵ.

accuracy	No.of $(H, \frac{\pi}{8})$ gates.
1	38
0.8	41
0.6	65
0.4	125
0.2	219
0.1	302
0.05	637
0.02	2525
0.01	5968
0.005	7496
0.001	28839
0.0001	306750
0.00001	3595036

For eg. when $U = \begin{bmatrix} 0.7631 - 0.4981j & 0.0080 - 0.4115j \\ -0.0080 - 0.4115j & 0.7631 + 0.4981j \end{bmatrix}$ and $\epsilon = 0.125$ we get,



So, approximating any arbitrary unitary requires huge number of gates as our accuracy decreases, nevertheless it provides a fault tolerant construction.

VI. Solovay-Kitaev Algorithm[3]

As we saw in the last section that approximating any arbitrary unitary matrix is hard as it requires large no of gates for smaller accuracy. For each matrix in $U(2)$ it requires $\mathcal{O}(\frac{1}{\epsilon})$ gates. However by solovay-kitaev algorithm we can achieve the same using $\mathcal{O}(\log^c\frac{1}{\epsilon})$ gates which significantly betters the synthesis of unitaries. We will take a brief overview into the algorithm.

Theorem 1: Let $\mathcal{G}$ be a finite subset of $SU(2)$ that is closed under inverses and $< \mathcal{G} >$ is dense in $SU(2)$, then there is a constant c,such that for any $\mathcal{G}$ and $\epsilon > 0$ one can choose $l = \mathcal{O}(\log^c\frac{1}{\epsilon})$ so that $\mathcal{G}^l$ is an $\epsilon - net$ for $SU(2)$.

The **sk algorithm**[3] was implement in Python3.6.

Algorithm 3 Solovay-Kitaev algorithm

function SK(Gate U,depth n)
  if $n == 0$ **then**
    return nn_Search(U)
  else
    $U_{n-1} \leftarrow SK(U, n-1)$
    $V, W \leftarrow$ Group_commutator_decompose(UU_{n-1})
    $V_{n-1} \leftarrow SK(V, n-1)$
    $W_{n-1} \leftarrow SK(W, n-1)$
    return $V_{n-1}W_{n-1}V_{n-1}^{\dagger}W_{n-1}^{\dagger}U_{n-1}$

We chose $\mathcal{G} = \{H, T, T^{\dagger}, S, S^{\dagger}\}$. Our main challenge was to carry out the basic approximation within accuracy $\frac{1}{32}$ so that the succeeding stages of the recursion converges the sequence of ϵ_k , where $\epsilon_k = \frac{1}{c^2}(c^2\epsilon)^{(3/2)^n}$, $c = 4\sqrt{2}$

Brute-force search was very slow for the basic-approximation in the base case of recursion, so we used kd-trees which has a fast nearest neighbor search! Any unitary matrix when converted to a matrix in $SU(2)$ neglecting the global phase factor has some interesting property w.r.t kd-trees.

$E(U, V) = ||U - V||_{trace-norm} = Tr\sqrt{(U-V)(U-V)^{\dagger}}$ $U, V \in SU(2)$

$$U = \begin{bmatrix} u_1 & u_2^* \\ -u_2 & u_1^* \end{bmatrix} V = \begin{bmatrix} v_1 & v_2^* \\ -v_2 & v_1^* \end{bmatrix} \quad (5)$$

$E(U - V) = \sqrt{2[(u_1 - v_1)^2 + (u_2 - v_2)^2]}$.Thus minimizing the euclidean norm of the vectors (u_1, u_2) and (v_1, v_2) suffices to state the nearest neighbor of the matrix U is V. Thus every matrix $U \in SU(2)$ is represented as a vector (u_1, u_2) in the kd-tree with the euclidean norm as its metric.

References

[1] Chi-Kwong Li,Rebecca Roberts and Xiaoyan Yin: Decomposition of unitary matrices and quantum gates.arXiv:quant-ph/1210.7366.

[2] A.Barenco,C.H.Bennet,R.Cleve,D.P.DiVincenzo,N.Margolus,P.Shor, T.Sleator,J.Smolin,H.Weinfurter: Elementary Gates for Quantum computation.arXiv:quant-ph/9503016.

[3] Michael A. Nielsen and Isaac L. Chuang:The Solovay Kitaev algorithm.arXiv:quant-ph/0505030.

[4] Maris Ozols:The Solovay-Kitaev algorithm.

[5] Michael A. Nielsen and Isaac L. Chuang:Quantum computing and Quantum information.