Converting the transitions between quantum gates into rotations

Nikolay Raychev

Abstract – This report describes an approach for generation of rotation through quantum operators. The approach of the proposed method transforms the transitions between quantum gates in rotary operations. Operations with qubits are very similar to the rotation, but with an added phase coefficient. This fact is used to create a process for rotation between unitary matrices.

Index Terms— boolen function, circuit, composition, encoding, gate, quantum.

1 Introduction

In the development process of a circuit quantum simulator [10, 11, 12] was required to seek solutions to approximate the results of the quantum operations, in order to animate what occurs without any sharp jumps. The fundamental concept for realization of the transitions is based on the fact that the quantum operation is always just a unitary matrix, which may be a linear interpolation between the matrices: $U_t = U_0(1-t) + U_1 t$. The operations with single qubits are very similar to the rotations, but with an added coefficient of the phase. This fact will be used to create a method for transformation of rotational into qubit operations.

The problem with the linear transformation is that the intermediate matrices may not be valid operations. The linear transformation tends to create a matrix entries, which are too close to zero, i.e. the resulting matrices will shrink the values instead of retaining their length (which is not a desired effect, as the whole point of using unitary matrices is to preserve the length). Actually, the goal is to be made a transformation without leaving the space of the unitary matrices. A compact way for the parameterization of the space of the unitary matrices is:

$$U = e^{\phi i}\left(Ii\cos(\theta) + \hat{v}\sigma_{xyz}\sin(\theta)\right)$$

The above equation includes four constants and three variables. The constants are the identity matrix (I), the square root of -1 (i), the constant of Euler (e) and the vector of Pauli matrices σ_{xyz}. The three variables are the angle ϕ, the angle θ and the single vector $\hat{v}$. Each of the variables plays a different role. ϕ is a global phase coefficient. It's what distinguishes the group of unitary matrices U(2) from "the special unitary group SU(2)". $\hat{v}$ and θ correspond to a rotation. $\hat{v}$ is like an axis to rotate around, and θ is how much to rotate around the said axis. What does it mean that $\hat{v}$ and θ are like a rotation? It becomes a bit clearer when the compact parameterization from above is expanded. Through incorporation of the Pauli matrices and splitting of $\hat{v}$ in $\langle x|y|z\rangle$ is obtained:

$$U = e^{\phi i}\left(i\cos(\theta)\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + x\sin(\theta)\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} + y\sin(\theta)\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix} \text{ и } z\sin(\theta)\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\right)$$

This is not yet clear enough. The following part is omitted: the equation to convert from an axis-angle style rotation to a single quaternion style rotation:

$$q = i\cos\left(\frac{\theta}{2}\right) + x\sin\left(\frac{\theta}{2}\right)i + y\sin\left(\frac{\theta}{2}\right)j + z\sin\left(\frac{\theta}{2}\right)k$$

The resemblance is visible. Without paying attention to the mysterious division by 2 of the angles, the Pauli matrices actually play the role of quaternion constants: i,j and k If each of the Pauli matrices is multiplied by i, is obtained:

$$(i\sigma_x)^2 = (i\sigma_y)^2 = (i\sigma_z)^2 = i^3\sigma_x\sigma_y\sigma_z = -I$$

This, in turn, looks very similar to the way in which the quaternions are defined: $i^2 = j^2 = k^2$=ijk = -1 Why is this similarity with the rotations important? Because it will be used for linear transformation. There are already existing methods for smooth linear transformation between quaternions and they will be applied in order to be handled the rotation part of the unitary operation. Then for the remaining phase part simply must be interpolated between two angles.

2 Converting the Transitions between Quantum Gates in Rotary Operations

First the unitary operation must be broken down into its quaternion and phased parts. Let's start by braking down the previous parameterization of the unitary group into a single matrix:

$$U = e^{\phi i}\begin{bmatrix} i\cos(\theta) + z\sin(\theta) & (x+iy)\sin(\theta)\\ (x-iy)\sin(\theta) & i\cos(\theta) - z\sin(\theta)\end{bmatrix}$$

The values for extraction are the phase ϕ and the quaternion components $i\cos(\theta)$, $x\sin(\theta)$, $y\sin(\theta)$ и $z\sin(\theta)$ It must be observed that $x\sin(\theta)$and $y\sin(\theta)$ contribute only for the upper right and lower left part of the matrix. In addition, xsin(θ)contributes symmetrically, while $y\sin(\theta)$ - asymmetrically. This allows to be solved their values, although they are still mixed with the phase, by taking the sum and the difference along the diagonal. The same applies for $z\sin(\theta)$ and $i\cos(\theta)$ along the other diagonal. To eliminate the coefficient $e^{\phi i}$ from the extracted values, is used the fact that it should be the only contributor of the complex values. Any component from the extracted four quaternion components can be selected (as long as it's not zero) and pick a phase coefficient, which will make the chosen component real. Since the given matrix certainly is unitary, the same coefficient of the phase should make the remaining quaternion components real. Below is given a code, written in python, which carries out the described factorization:

```
def quantum_unitary_breakdown(m):
    Breaks an unitary matrix in quaternion and phase components.

    # Extract rotation components
    a, b, c, d = m[0, 0], m[0, 1], m[1, 0], m[1, 1]
    t = (a + d)/2j
    x = (b + c)/2
    y = (b - c)/-2j
    z = (a - d)/2
```

```
  # Extracts common phase coefficient
  p = max([t, x, y, z], key=lambda is: abs(e))
  p /= abs(p)
  pt, px, py, pz = t/p, x/p, y/p, z/p

  q = [pt.real, px.real, py.real, pz.real]
  return q, p
```

After the problem can be broken down into factors in the rotation and phase parts, and they can be interpolated separately. For the rotation part will be used a spherical transformation. In order to be interpolated spherically between two points - p_0 andp_1 must be found an angle, satisfying $\cos(\theta) = p_0 \cdot p_1$, and then the result is:

$$SLerpolation(p_0, p_1, t) = \frac{\sin(\theta(1-t))}{\sin(\theta)} p_0 + \frac{\sin(\theta t)}{\sin(\theta)} p_1$$

The obstacle here is the division by zero, when θ is zero. Fortunately, because the numerator is approaching zero generally in the same way as the denominator, this is a case in which the obtained value does not deviate. A function can be defined, which calculates $\frac{\sin(xf)}{\sin(x)}$, but switches to an approximation, that does not divide by zero or increase the errors at floating point numbers, when they are close to zero:

def quantum_sin_scale_ratio(theta, factor):
Returns sin(theta * factor) / sin(theta) with care around the origin to avoid dividing by zero.

```
# Near zero, transition to an approximation, to avoid an increase
from error at floating point numbers.
  if abs(theta) < 0.0001:
    # sin(x) = x - x^3/3! + ...
         # sin(f x) / sin(x)
         # = ((fx) - (fx)^3/3! + ...) / (x - x^3/3! + ...)
         # ~= ((fx) - (fx)^3/3!) / (x - x^3/3!)
         # = (f - f(fx)^2/3!) / (1 - x^2/3!)
         # = f (1 - f^2 x^2/6) / (1 - x^2/6)
    d = theta * theta / 6
    return factor * (1 - d * factor * factor) / (1 - d)
  return math.sin(theta * factor) / math.sin(theta)
```

The above method will be applied at the method for full transformation, when a spherical transformation is being carried out. In order to make an angular interpolation the obvious shall be carried out: the difference between the two angles is learned, care should be taken to recourse to a roundabout way and then a linear transformation should be made. To take correctly the sign of the difference is a difficult task, but it is already explained. When everything is put together, we obtain:

def quantum_unitary_lerp(u1, u2, t):
Interpolates between two 2x2 unitary NumPy matrices.

```
  # Split into rotation and phase parts
  q1, p1 = quantum_unitary_breakdown(u1)
  q2, p2 = quantum_unitary_breakdown(u2)
  # Spherical transformation of the rotation
  dot = sum(v1*v2 for v1,v2 in zip(q1, q2))
  if dot < 0:
    # Not to be made in the long way around...
    q2 *= -1
    p2 *= -1
    dot *= -1
  theta = math.acos(min(dot, 1))
  c1 = sin_scale_ratio(theta, 1-t)
  c2 = sin_scale_ratio(theta, t)
  u3 = (u1 * c1 / p1 + u2 * c2 / p2)
  # Angular transformation of the phase
  a1 = np.angle(p1)
  a2 = np.angle(p2)
  # The smallest angular distance with a sign (mod 2pi)
  da = (a2 - a1 + math.pi) % (math.pi * 2) - math.pi
  a3 = a1 + da * t
  p3 = math.cos(a3) + 1j * math.sin(a3)
  return u3 * p3
```

Demonstration
Below is given a demonstration of the described above method with the developed by the author of the article quantum simulator. Matrices for start and end can be entered in the text boxes and (after the entered matrices are adjusted, they should be unitary) a continuous transition between the two matrices is shown. It is difficult to be checked ostensibly, whether the intermediate matrices are unitary, but it can be seen that the movement is smooth and the colored area remains relatively constant.

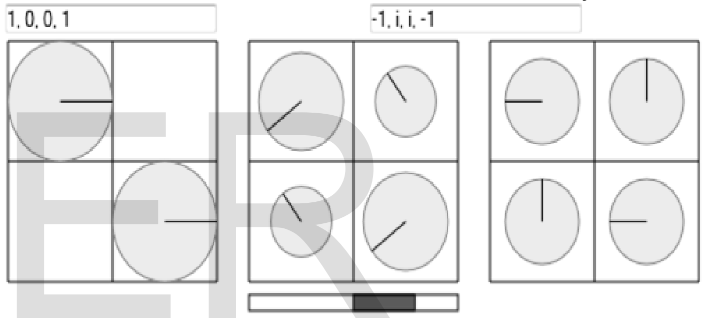

Figure 1: Linear transformation

(Note: The input correction is done by doing a singular value decomposition and omitting the non-unitary factor. This turns out to be really effective.).

3 Conclusion

Single qubit operations are a lot like rotations, but with an added phase coefficient. This fact can be used to create a method for linear transformation between 2 unitary matrices. The method described above, works, but is not optimal. For example, it does not ensure a constant angular speed. Also, in some cases it doesn't take the shortest possible path.

REFFERENCES
[1] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information (Cambridge University Press, Cambridge, United Kingdom, 2000).
[2] I. L. Chuang, N. Gershenfeld and M. Kubinec, "Experimental Implementation of Fast Quantum Searching", Physical Review Letters, 80(15), 3408-3411, 1998.
[3] R. Cleve and J. Watrous, "Fast Parallel Circuits for the Quantum Fourier Transform", Proceedings of IEEE Symposium on the Theory of Computing, pp. 526-535, 2000.
[4] W. Cooley and J. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series", Math. Of Computation, 19:297-

301, 1965.
[5] D. Coppersmith, "An Approximate Fourier Transform Useful in Quantum Factoring", IBM Research Report RC 19642, 1994.
[6] X. Deng, T. Hanyu and M. Kameyama, "Quantum Device Model Based Super Pass Gate for Multiple-Valued Digital Systems", Proceedings of Intl. Symp. Multiple-Valued Logic, pp. 130-138, 1995.
[7] D. Deutsch, "Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer", Proc. Royal Soc. A 400:97-117, 1985.
[8] N. Gershenfeld and I. L. Chuang, "Bulk Spin-Resonance Quantum Computing", Nature, vol. 404, pp.350-356, 1997.
[9] R. Duan, Z. Ji, Y. Feng and M. Ying, "A Relation Between Quantum Operations and the Quantum Fourier Transform", Quantum Physics Archive, arxiv: quant-ph/0304145
[10] L. Hales, "Quantum Fourier Transform and Extensions of the Abelian Hidden Subgroup Problem", Ph. D. Dissertation, UC Berkeley, 2002.
[11] S. L. Hurst, D. M. Miller and J. Muzio, "Spectral Techniques in Digital Logic", Academic Press, London, 1985.
[12] R. Jozsa, "Quantum Algorithms and the Fourier Transform", Proceedings of Royal Society of London, 454:323-337, 1997.
[13] Nikolay Raychev. Dynamic simulation of quantum stochastic walk. In International jubilee congress (TU), 2012.
[14] Nikolay Raychev. Classical simulation of quantum algorithms. In International jubilee congress (TU), 2012.
[15] Nikolay Raychev. Interactive environment for implementation and simulation of quantum algorithms. CompSysTech'15, DOI: 10.13140/RG.2.1.2984.3362, 2015
[16] Nikolay Raychev. Unitary combinations of formalized classes in qubit space. In International Journal of Scientific and Engineering Research 04/2015; 6(4):395-398, 2015.
[17] Nikolay Raychev. Functional composition of quantum functions. In International Journal of Scientific and Engineering Research 04/2015; 6(4):413-415, 2015.
[18] Nikolay Raychev. Logical sets of quantum operators. In International Journal of Scientific and Engineering Research 04/2015; 6(4):391-394, 2015.
[19] Nikolay Raychev. Controlled formalized operators. In International Journal of Scientific and Engineering Research 05/2015; 6(5):1467-1469, 2015.
[20] Nikolay Raychev. Controlled formalized operators with multiple control bits. In International Journal of Scientific and Engineering Research 05/2015; 6(5):1470-1473, 2015.
[21] Nikolay Raychev. Connecting sets of formalized operators. In International Journal of Scientific and Engineering Research 05/2015; 6(5):1474-1476, 2015.
[22] Nikolay Raychev. Indexed formalized operators for n-bit circuits. In International Journal of Scientific and Engineering Research 05/2015; 6(5):1477-1480, 2015.