Christian B. Mendl, Martina Nibbi, Pedro Hack, Irene López Gutiérrez due: 03 Jul 2023, 08:00 on Moodle

Tutorial 10 (Block encoding techniques[1])

Qubitization consists of two fundamental subroutines: block encoding and signal processing. In this tutorial, we discuss strategies for block encoding, for which there are various problem-specific methods available.

The goal is to realize a linear map $\mathcal{H} \in \mathbb{C}^{N\times N}$ with spectral norm $\|\mathcal{H}\| \le 1$ by a quantum circuit, with $N = 2^n$ for n qubits. Block encoding refers to embedding $\mathcal{H}$ into a larger unitary matrix U, which is necessary since $\mathcal{H}$ is not unitary in general. Specifically, we use additional ancillary qubits, such that U maps $\mathbb{C}^K \otimes \mathbb{C}^N \to \mathbb{C}^K \otimes \mathbb{C}^N$, with $K = 2^k$ for k ancillary qubits. The actual encoding is then realized using a special ancillary state $|G\rangle \in \mathbb{C}^K$, such that

$$\mathcal{H} = (\langle G| \otimes I_N)\, U\, (|G\rangle \otimes I_N),$$

where I_N refers to the $N \times N$ identity matrix. For simplicity, we take $|G\rangle = |0,\dots,0\rangle$ in the following. The matrix representation of U is then of the form

$$U = \begin{pmatrix} \mathcal{H} & * \\ * & * \end{pmatrix},$$

where the stars refer to unspecified blocks.

(a) Find a block encoding for $\mathcal{H}$ Hermitian and with spectral norm $\|\mathcal{H}\| \le 1$ using only a single ancillary qubit.

The construction for U so far is general, but still leaves open the question of how to express U in terms of elementary gates. As an example for a more explicit circuit construction, we investigate the case that $\mathcal{H}$ is a linear combination of unitaries (LCU) in the following. Such a decomposition always exists (but might contain many terms): for example, one can use the set of Pauli strings as basis for the LCU.

(b) Consider the following Hamiltonian written as a sum of Pauli strings:

$$\mathcal{H} = \frac{1}{4}\big(XX + YZ + YY + ZX\big).$$

Construct a block encoding for $\mathcal{H}$ using elementary quantum circuit gates.

(c) Generalize the approach in (b), assuming that there exists a LCU decomposition of the Hamiltonian of the form

$$\mathcal{H} = \sum_{i=0}^{m-1} \alpha_i V_i. \tag{1}$$

Here each V_i is a unitary matrix, and without loss of generality, we can assume that the coefficients α_i are positive numbers, since phase factors can be absorbed into the unitaries. How does the number of ancillary qubits scale?

Exercise 10.1 (Numerical simulation of block encoding)

In this exercise, we programmatically realize the block encoding techniques discussed in the tutorial.

(a) Write a Python function which constructs

$$U = \begin{pmatrix} \mathcal{H} & \sqrt{I - \mathcal{H}^2} \\ \sqrt{I - \mathcal{H}^2} & -\mathcal{H} \end{pmatrix} \tag{2}$$

for a given Hermitian matrix $\mathcal{H}$. To test your implementation, first sample a random 3×3 complex Hermitian matrix $\mathcal{H}$, and rescale $\mathcal{H}$ such that all eigenvalues are in the interval $[-1, 1]$. Use your code to compute the corresponding U, and verify numerically that U is unitary and contains $\mathcal{H}$ in its upper left block.

Hint: The representation $U = Z \otimes \mathcal{H} + X \otimes \sqrt{I - \mathcal{H}^2}$ and `scipy.linalg.sqrtm` for the matrix square root might be helpful.

(b) Inserting the spectral decomposition $\mathcal{H} = \sum_j \lambda_j |\psi_j\rangle\langle\psi_j|$ (with eigenvalues $\lambda_j \in [-1, 1]$) into Eq. (2) leads to

$$U = \sum_j \begin{pmatrix} \lambda_j & \sqrt{1-\lambda_j^2} \\ \sqrt{1-\lambda_j^2} & -\lambda_j \end{pmatrix} \otimes |\psi_j\rangle\langle\psi_j| = \sum_j R(\lambda_j) \otimes |\psi_j\rangle\langle\psi_j|, \quad R(a) = \begin{pmatrix} a & \sqrt{1-a^2} \\ \sqrt{1-a^2} & -a \end{pmatrix}.$$

[1] G. H. Low and I. L. Chuang: *Hamiltonian simulation by qubitization.* Quantum 3, 163 (2019), and L. Lin: *Lecture notes on quantum algorithms for scientific computation.* `https://math.berkeley.edu/~linlin/qasc/`

hw10

July 1, 2023

[ ]:
```python
# Ex 10.1
# a)
import numpy as np
import scipy


X = np.array([[0, 1],[1, 0]])
Y = np.array([[0, -1j],[1j, 0]])
Z = np.array([[1, 0],[0, -1]])


def U(H):
        A = np.outer(Z, H)
        Id = np.eye((int)(np.sqrt(H.size)))
        B = scipy.linalg.sqrtm(Id - np.dot(H, H))
        C = np.outer(X, B)
        return A + C

H = np.array([[1, 0],[0, 1]])

phi = np.array([[1, 0, 0],[0, 1, 0], [0, 0, 1]])
# print(phi[0].T)
eigen_states = []
for i in phi:
        eigen_states.append(np.outer(i.T, i))

R = np.array([1, 0.5, 0.1])
H = R[0]*eigen_states[0] + R[1] *eigen_states[1] + R[2]*eigen_states[2]
print(H)
print(U(H))
```

```
[[1.  0.  0. ]
 [0.  0.5 0. ]
 [0.  0.  0.1]]
[[ 1.          0.          0.          0.          0.5         0.
   0.          0.          0.1       ]
 [ 0.          0.          0.          0.          0.8660254   0.
   0.          0.          0.99498744]
```

```
 [ 0.          0.          0.          0.          0.8660254   0.
   0.          0.          0.99498744]
 [-1.          0.          0.          0.         -0.5         0.
   0.          0.         -0.1       ]]
```

[ ]:
```
# Ex 10.1
# b)
phi = np.array([[1, 0, 0],[0, 1, 0], [0, 0, 1]])
# print(phi[0].T)
eigen_states = []
for i in phi:
        eigen_states.append(np.outer(i.T, i))

H_second = np.zeros((4,9))
for i in range(3):
        H_second += np.outer(U(R[i]), eigen_states[i])

print(H_second)
```

```
[[ 1.          0.          0.          0.          0.5         0.
   0.          0.          0.1       ]
 [ 0.          0.          0.          0.          0.8660254   0.
   0.          0.          0.99498744]
 [ 0.          0.          0.          0.          0.8660254   0.
   0.          0.          0.99498744]
 [-1.          0.          0.          0.         -0.5         0.
   0.          0.         -0.1       ]]
```

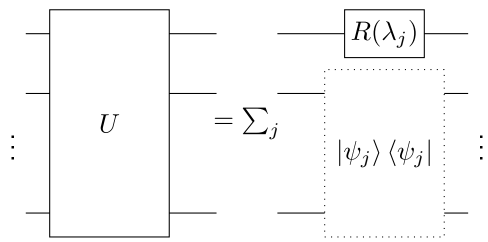
[ ]:
```
print(U(H) == H_second)
```

```
[[ True  True  True  True  True  True  True  True  True]
 [ True  True  True  True  True  True  True  True  True]
 [ True  True  True  True  True  True  True  True  True]
 [ True  True  True  True  True  True  True  True  True]]
```

[ ]:
```

```

A circuit diagram representation of this equation is given by



Verify that $R(\lambda_j) \otimes |\psi_j\rangle\langle\psi_j|$ agrees with U from (a) (for the same random Hamiltonian matrix $\mathcal{H}$).

(c) Write a Python/NumPy program to simulate the block encoding method in tutorial 10 (c) for a linear combination of unitaries (LCU) in Eq. (1) with $m = 4$ and $\|\alpha\|_1 = 1$. You can realize each controlled-V_i gate in the SELECT operation via

$$\text{controlled-}V_i = |i\rangle\langle i| \otimes V_i + (I_m - |i\rangle\langle i|) \otimes I_N,$$

where $|i\rangle$ is the i-th unit vector in $\mathbb{R}^m$. The second summand corresponds to inactive control. To construct the PREPARE unitary matrix, you can start from a $m \times 1$ matrix containing $\sqrt{\alpha}$ as its single column, and then use a "complete" QR-decomposition (`np.linalg.qr`) to extend it to a $m \times m$ matrix. The resulting Q matrix could contain $-\sqrt{\alpha}$ (instead of $\sqrt{\alpha}$) in its first column; in this case, use $-Q$ (instead of Q) as PREPARE gate.

To test your implementation, first draw four random unitary 3×3 matrices $\{V_i\}$ (Haar random distribution) and corresponding random non-negative coefficients α_i, normalized such that $\|\alpha\|_1 = 1$. Then compute the matrix representation of the circuit from tutorial 10 (c), and verify that it is unitary and contains $\mathcal{H}$ defined in Eq. (1) in its upper left 3×3 block.

Hint: You can use `scipy.stats.unitary_group` to sample a Haar random unitary matrix.

Exercise 10.2 (General signal-processing operation)

The quantum eigenvalue transformation combines the block encoding of a Hamiltonian $\mathcal{H}$ with quantum signal processing. In case of a single auxiliary qubit for block encoding as in tutorial 10 (a), the corresponding quantum circuit is specified by (with $\vec{\varphi} \in \mathbb{R}^{d+1}$)

$$U_{\vec{\varphi}} = \mathrm{e}^{i\varphi_0 Z} \cdots U^\dagger \, \mathrm{e}^{i\varphi_{d-1} Z} \, U \, \mathrm{e}^{i\varphi_d Z},$$

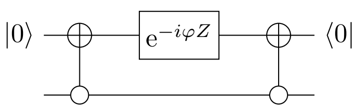
where U is the block encoding gate, the alternation between U and $U^\dagger$ is analogous to the amplitude amplification example, and the signal-processing gates $\mathrm{e}^{i\varphi_k Z}$ act on the auxiliary qubit. Note the representation $\mathrm{e}^{i\varphi Z} = \mathrm{e}^{i\varphi(2|0\rangle\langle 0| - I)}$. For a general block encoding gate, the signal-processing phase operation generalizes to (for $\varphi \in \mathbb{R}$)

$$\Pi_\varphi = \mathrm{e}^{i\varphi(2\Pi - I)},$$

where Π is the projector onto the block in U containing $\mathcal{H}$. In the context of tutorial 10, $\Pi = |G\rangle\langle G| \otimes I_N$. In this exercise, our goal is to find quantum circuits realizing such signal-processing operations.

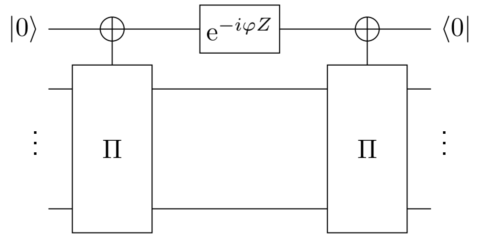
(a) Prove that the following circuit (using an additional auxiliary qubit) implements the projector-controlled phase operation for $\Pi = |0\rangle\langle 0|$:

$\mathrm{e}^{i\varphi(2|0\rangle\langle 0| - I)} =$



(b) A circuit construction for a general projector Π is shown below, where the control is activated by states from the subspace which Π projects to. Argue why this indeed implements Π_φ:

$\mathrm{e}^{i\varphi(2\Pi - I)} =$

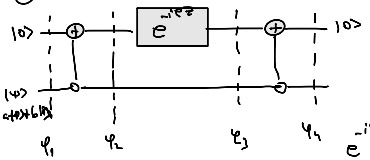


(c) Explicitly specify the circuit from (b) for $\Pi = |G\rangle\langle G|$ with $|G\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ using only elementary gates.

Hint: Think about a circuit that transforms $|G\rangle \mapsto |00\rangle$. You are allowed to use a multi-controlled NOT gate as elementary gate.

Ex 10.2.

(a).



$$e^{-i\varphi Z} = \cos(\varphi)\mathbb{1} - iZ\sin(\varphi) = \begin{pmatrix} e^{-i\varphi z} & 0 \\ 0 & e^{+i\varphi z} \end{pmatrix}$$

$$\varphi_1 = |0\rangle \otimes (a|0\rangle + b|1\rangle)$$

$$\varphi_2 = a|00\rangle + b|11\rangle$$

$$\varphi_3 = a \cdot e^{-i\varphi}|0\rangle \otimes |0\rangle + b \cdot e^{i\varphi}|1\rangle \otimes |1\rangle$$

$$\varphi_4 = a \cdot e^{-i\varphi}|0\rangle \otimes |0\rangle + b e^{i\varphi}|0\rangle \otimes |1\rangle$$

$$= |0\rangle \otimes (a e^{-i\varphi}|0\rangle + b e^{i\varphi}|1\rangle)$$

$$= |0\rangle \otimes \left(\begin{pmatrix} e^{-i\varphi} & 0 \\ 0 & e^{i\varphi} \end{pmatrix} \cdot (a|0\rangle + b|1\rangle) \right)$$

$$= |0\rangle \otimes \left(e^{i\varphi Z} (a|0\rangle + b|1\rangle) \right)$$

$$= |0\rangle \otimes e^{i\varphi(2|0\rangle\langle 0| - \mathbb{1})} \cdot (a|0\rangle + b|1\rangle)$$

(b). Π_φ!