Christian B. Mendl, Irene López Gutiérrez, Keefe Huang due: 13 Jan 2021

Tutorial 8 (Classical Fourier transformation)
We use the following convention for the discrete Fourier transform:

$$\mathcal{F} : \mathbb{C}^N \to \mathbb{C}^N, \quad \mathcal{F}(x) = y \quad \text{with} \quad y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \, \mathrm{e}^{2\pi i jk/N} \quad \text{for} \quad k = 0, \dots, N-1.$$

(a) Show that $\mathcal{F}$ is a unitary operation, i.e., its matrix representation

$$U_{\mathcal{F}} = (u_{kj}), \quad u_{kj} = \frac{1}{\sqrt{N}} \mathrm{e}^{2\pi i jk/N}$$

is a unitary matrix.

(b) Write down the inverse Fourier transform based on part (a).

(c) The *discrete circular convolution* $*$ of two vectors $x, y \in \mathbb{C}^N$ yields another vector and is defined as

$$(x * y)_j = \sum_{\ell=0}^{N-1} x_\ell \, y_{(j-\ell) \bmod N} \quad \text{for} \quad j = 0, \dots, N-1. \tag{1}$$

Derive the *circular convolution theorem*: for any $x, y \in \mathbb{C}^N$,

$$x * y = \sqrt{N} \, \mathcal{F}^{-1}\big(\mathcal{F}(x) \cdot \mathcal{F}(y)\big),$$

where $\cdot$ denotes pointwise multiplication. Also compare the asymptotic runtime when using the FFT algorithm for the Fourier transforms, as compared to a literal implementation based on the definition (1).

Exercise 8.1 (Decomposition of controlled-U gates)
Recall the *Z-Y decomposition* theorem from exercise 2.1: given any unitary 2×2 matrix U, there exist real numbers $\alpha, \beta, \gamma, \delta \in \mathbb{R}$ such that

$$U = \mathrm{e}^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta), \tag{2}$$

where R_y, R_z are the rotation operators

$$R_y(\theta) = \mathrm{e}^{-i\theta Y/2} = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, \qquad R_z(\theta) = \mathrm{e}^{-i\theta Z/2} = \begin{pmatrix} \mathrm{e}^{-i\theta/2} & 0 \\ 0 & \mathrm{e}^{i\theta/2} \end{pmatrix}.$$

Via this theorem, we will construct a circuit solely consisting of CNOT and single unitary gates to implement a controlled-U operation.

(a) Show that $XYX = -Y$, $XZX = -Z$ and thus $XR_y(\theta)X = R_y(-\theta)$ and $XR_z(\theta)X = R_z(-\theta)$, where X, Y and Z are the usual Pauli matrices.

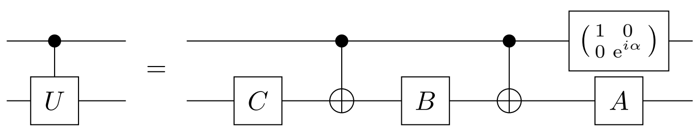
(b) Based on the Z-Y decomposition in Eq. (2), we define the unitary operators

$$A = R_z(\beta) R_y(\tfrac{1}{2}\gamma), \quad B = R_y(-\tfrac{1}{2}\gamma) R_z(-\tfrac{1}{2}(\delta + \beta)), \quad C = R_z(\tfrac{1}{2}(\delta - \beta)).$$

Verify that $ABC = I$ and $U = \mathrm{e}^{i\alpha} AXBXC$.

Hint: Insert $X^2 = I$ in the "center" of XBX and use part (a).

(c) Argue that the following circuit implements the controlled-U operation, with the definitions in part (b):

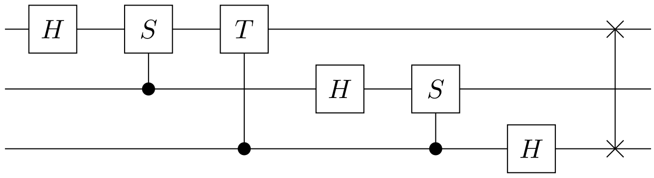


Hint: Distinguish whether the control qubit is set to $|0\rangle$ or $|1\rangle$.

(d) Decompose the controlled-R_k gate appearing in the quantum Fourier transform into single qubit and CNOT gates.

Hint: It is convenient to choose $\beta = 0$ and $\gamma = 0$ in Eq. (2) for decomposing $R_k = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{2\pi i/2^k} \end{pmatrix}$.

Exercise 8.2 (Three qubit quantum Fourier transform implementation)
The explicit circuit for the three qubit quantum Fourier transform is



Besides the Hadamard gate H, the single qubit gates appearing in the circuit are:

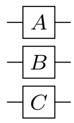
$$\text{phase gate} \quad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \qquad \pi/8 \text{ gate} \quad T = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{i\pi/4} \end{pmatrix}.$$

You should convince yourself that S and T are special cases of the rotation operators R_k, namely $S = R_2$ and $T = R_3$.

Construct the matrix representation of this circuit using Python/NumPy, and verify that it indeed agrees with the Fourier transform matrix.

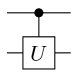
Hints and suggestions:

- One can obtain the matrix representation of the circuit shown on the right via `np.kron(np.kron(A, B), C)`. This can be useful, e.g., for the Hadamard gates appearing in the quantum Fourier circuit (by setting two of the A, B, C matrices to 2×2 identity matrices).



- We have already encountered the matrix representation of a controlled-U gate:

$$\begin{pmatrix} 1 & & \\ & 1 & \\ & & U \end{pmatrix},$$



  where the empty fields are all 0 and U occupies the lower right 2×2 block.

- The matrix representation of the swap gate was the topic of exercise 2.1 (a).
- It will be necessary to apply a given gate for a specific ordering of the qubits. E.g., considering the controlled-T gate in the quantum Fourier circuit, the *third* qubit is the control and the *first* qubit the target. Such a reordering can be achieved by the following utility function:

```
def reorder_gate(G, perm):
    """
    Adapt gate 'G' to an ordering of the qubits as specified in 'perm'.

    Example, given G = np.kron(np.kron(A, B), C):
        reorder_gate(G, [1, 2, 0]) == np.kron(np.kron(B, C), A)
    """
    perm = list(perm)
    # number of qubits
    n = len(perm)
    # reorder both input and output dimensions
    perm2 = perm + [n + i for i in perm]
    return np.reshape(np.transpose(np.reshape(G, 2*n*[2]), perm2), (2**n, 2**n))
```

  As illustration, `reorder_gate(np.kron(controlled_gate(T), np.identity(2)), [1, 2, 0])` then constructs the controlled-T gate appearing in the quantum Fourier circuit as 8×8 matrix, where `controlled_gate(U)` is the 4×4 matrix representation of a controlled-U gate as shown above.

- The Fourier transform matrix to be used as reference can be assembled via

```
np.array([[np.exp(2*np.pi*1j*j*k/8)/np.sqrt(8) for j in range(8)] for k in range(8)])
```