12

Quantum Fourier transform

The authors are grateful to Ronald de Wolf for reviewing this chapter.

Rough overview (in words)

The quantum Fourier transform (QFT) is a quantum version of the discrete Fourier transform (DFT) and takes quantum states to their Fourier-transformed version.

Rough overview (in math)

The QFT is a quantum circuit that takes pure N-dimensional quantum states $|x\rangle = \sum_{i=0}^{N-1} x_i|i\rangle$ to pure quantum states $|y\rangle = \sum_{i=0}^{N-1} y_i|i\rangle$ with the Fourier-transformed amplitudes

$$y_k = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} x_l \exp(2\pi \mathrm{i} kl/N) \quad \text{for } k = 0, \ldots, N-1. \tag{12.1}$$

Dominant resource cost (gates/qubits)

The space cost is $O(\log(N))$ qubits and the quantum complexity of the textbook algorithm is $O(\log^2(N))$. In terms of Hadamard gates, swap gates, and controlled phase shift gates $|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes R_\ell$ with

$$R_\ell = \begin{pmatrix} 1 & 0 \\ 0 & \exp\left(2\pi \mathrm{i} 2^{-\ell}\right) \end{pmatrix},$$

the quantum circuit is given in Fig. 12.1 (see also [801, Fig. 5.1]), where $N = 2^n$. The swap gates at the end of the circuit are required to reverse the order of the output qubits. The complexity can be improved to

$$O\left(\log(N) \log\left(\log(N)\epsilon^{-1}\right) + \log^2\left(\epsilon^{-1}\right)\right)$$

https://doi.org/10.1017/9781009639651.015 Published online by Cambridge University Press

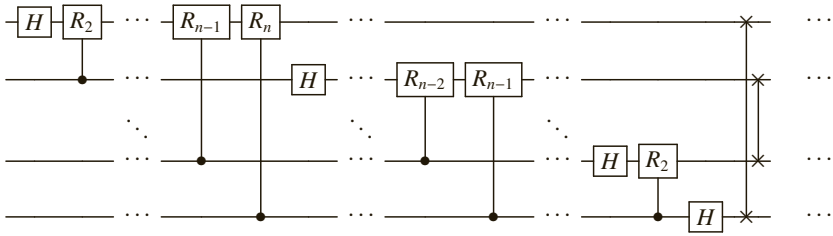


Figure 12.1 Quantum circuit implementation of QFT.

when only asking for ϵ-approximate solutions [486]. Finite constants and compilation cost for fault-tolerant quantum architectures are also discussed in the literature. For example, [795] gives an implementation with $O(\log(N)\log\log(N))$ T gates and estimates finite T-gate costs for different instance sizes.

Caveats

- The QFT does not achieve the same task as the classical DFT that takes vectors $(x_0, \ldots, x_{N-1}) \in \mathbb{C}^N$ to vectors $(y_0, \ldots, y_{N-1}) \in \mathbb{C}^N$ with y_k defined as in Eq. (12.1). The DFT can be implemented via the fast Fourier transform in classical complexity $O(N\log(N))$, which is exponentially more costly than the quantum complexity $O(\log^2(N))$ of the QFT. However, for the QFT to achieve the same task as the DFT, pure state quantum tomography would be required to read out and learn the Fourier-transformed amplitudes, which destroys any quantum speedup for the DFT.
- When QFT is employed in use cases, for example, for factoring, one has to be careful in finite-size instances when counting resources [943], and for this a semiclassical version of the QFT can be more quantum resource efficient [458].
- The QFT admits an efficient representation as a matrix product operator (a type of tensor network), meaning that the approximation improves exponentially in the bond dimension [262]. This suggests that quantum algorithms relying on the QFT for speedup must involve highly entangled input or intermediate states, in order to beat state-of-the-art tensor network methods.

Example use cases

- Even though the QFT does not speedup the DFT, QFT is used as a subroutine in more involved quantum routines with large quantum speedup. Examples include quantum algorithms for the discrete logarithm problem, the

https://doi.org/10.1017/9781009639651.015 Published online by Cambridge University Press

hidden subgroup problem, the factoring problem, to name a few. The QFT can be seen as the crucial quantum ingredient that allows for a superpolynomial end-to-end quantum speedup for these problems. We discuss this in the context of quantum cryptanalysis in Chapter 6.

- The QFT appears in the standard circuit for quantum phase estimation, where it is used to convert accrued phase estimation into a binary value that can be read out.
- The QFT is used for switching between the position and momentum bases in grid-based simulations of quantum chemistry [601] or quantum field theories [588].

Further reading

- Textbook reference [801, Section 5.1].
- The quantum Fourier transform can be generalized to other groups. The version presented above is for the group $\mathbb{Z}/(2^n\mathbb{Z})$. Its implementation for other abelian groups as well as nonabelian groups is discussed in [277] and the references therein.

https://doi.org/10.1017/9781009639651.015 Published online by Cambridge University Press