Explaining quantum phase estimation using data structures

Shashvat Shukla

October 1, 2025

Abstract

The quantum phase estimation algorithm is an important technique that is applied in physical simulation and prime factorisation on quantum computers. Most existing presentations of quantum phase estimation use an algebraic manipulation to show that the quantum circuit for phase estimation correctly performs what it is supposed to. However this can obscure the overarching algorithmic idea. This paper explains the phase estimation algorithm in a new way, using elementary polynomial data structures from computer science as helpful high-level concepts. Once the polynomial data structure concepts have been understood, the explanation of quantum phase estimation can be stated in just a few lines. We hope this explanation will provide instructors with an alternative as quantum computing grows in popularity.

1 Introduction

Quantum computers promise new ways of solving problems that are intractable on classical computers. Their power comes from quantum bits (qubits), which, unlike classical bits, can exploit superposition, interference, and entanglement to store and process information more efficiently. Among the most important algorithms that harness this power is quantum phase estimation, a cornerstone of both algorithms for the simulation of physical systems and Shor's factoring algorithm.

Most existing presentations of phase estimation are mathematically detailed. They generally present the quantum circuit for the algorithm and present an analysis of the correctness of the algorithm by means of algebraic manipulation. While rigorous, they often obscure the underlying ideas and can be unwieldy in settings where a quick, clear explanation is all that is needed. The usefulness of this paper is in providing an alternative explanation: by framing phase estimation in terms of high level algorithmic concepts, we show how elementary data structures for representing and multiplying polynomials efficiently and a divide-and-conquer approach can help explain at a high level what the quantum phase estimation algorithm is doing.

As quantum computing is a growing scientific area and industry in its own right, and as more branches of physics incorporate quantum computational methods, an ever-larger community of physicists will need at least a working understanding of phase estimation. With the rapid growth of undergraduate programs in quantum computing, it is no longer just quantum information theorists who encounter this algorithm; many instructors across physics will find themselves teaching or applying it. A clear, adaptable account is therefore of broad and growing value, both for students entering the field and for physicists in other areas who will increasingly meet quantum computing in their research.

2 Polynomial data structures and the quantum Fourier transform

This section introduces the key data structure concepts that we use in our explanation of quantum phase estimation. Figure 1 shows these concepts and how they are related in one graphic. Once the concepts in this section have been understood, then QPE can be explained in just a few lines as we do in the next subsection.

2.1 Polynomial data structure concepts

The coefficient representation of a polynomial [1] is a vector containing all its coefficients. For example $p(x) = a_{N-1}x^{N-1} + ... + a_0$ is represented as $a = [a_0...a_{N-1}]$.

The point-value representation of a polynomial [1] is motivated by the fact that a polynomial (p with degree less than N) is represented by a series of points $\{(x_0, p(x_0)), ..., (x_{N-1}, p(x_{N-1}))\}$. For example, with two points, we represent the unique line that passes through those points. If we fix the inputs $\{x_0, ..., x_{N-1}\}$ at which we will evaluate the polynomial, then we can just store the values $[p(x_0)...p(x_{N-1})]$ as a vector. We can choose any set of N input values. For our purposes we will choose the N-th roots of unity $\{1, \omega_N, ..., \omega_N^{N-1}\}$ as our set of inputs and store $[p(1)...p(\omega_N^{N-1})]$. This choice makes it efficient to convert between the coefficient and point-value representations.

Consider how we can convert from the coefficient representation to the point value representation. We have to evaluate the polynomial at each of the N inputs. Converting $[1, \omega_N, ..., \omega_N^{N-1}]$ to $[p(1), ..., p(\omega_N^{N-1})]$ requires the following transformation:

$$p(\omega_N^l) = \sum_{j=0}^{N-1} a_j \omega_N^{jl} = \sum_{j=0}^{N-1} a_j e^{i2\pi jl/N},$$

which is exactly the Discrete Fourier Transform.

2.2 Quantum Encoding of polynomial data structures

The amplitude encoding of a vector is the quantum state obtained from normalising the components of a vector and putting them into amplitudes of a quantum state, for example as: $v = [v_0 ... v_{N-1}] \rightarrow \frac{1}{\|v\|} \sum_{i=0}^{N-1} v_i |i\rangle$.

When normalised by a $\frac{1}{\sqrt{N}}$ factor, the Discrete Fourier Transform is a unitary matrix. If we implement this matrix as a quantum circuit, it converts from the *amplitude encoding* of the coefficient representation to the *amplitude encoding* of the point-value representation. This circuit is called the Quantum Fourier Transform and it can be implemented efficiently due to a divide-and-conquer algorithm. The details of this are interesting but not essential for a first understanding of QPE, so we discuss it in the next section.

The final step of QPE involves the *amplitude encoding* of the coefficient representation of a polynomial so $p(x) = a_{N-1}x^{N-1} + ... + a_0$ is stored as $\frac{1}{\|a\|} \sum_{i=0}^{N-1} a_i |i\rangle$. Since the polynomial in question turns out to be a single term polynomial x^k, the amplitude encoding of it is just $|k\rangle$.

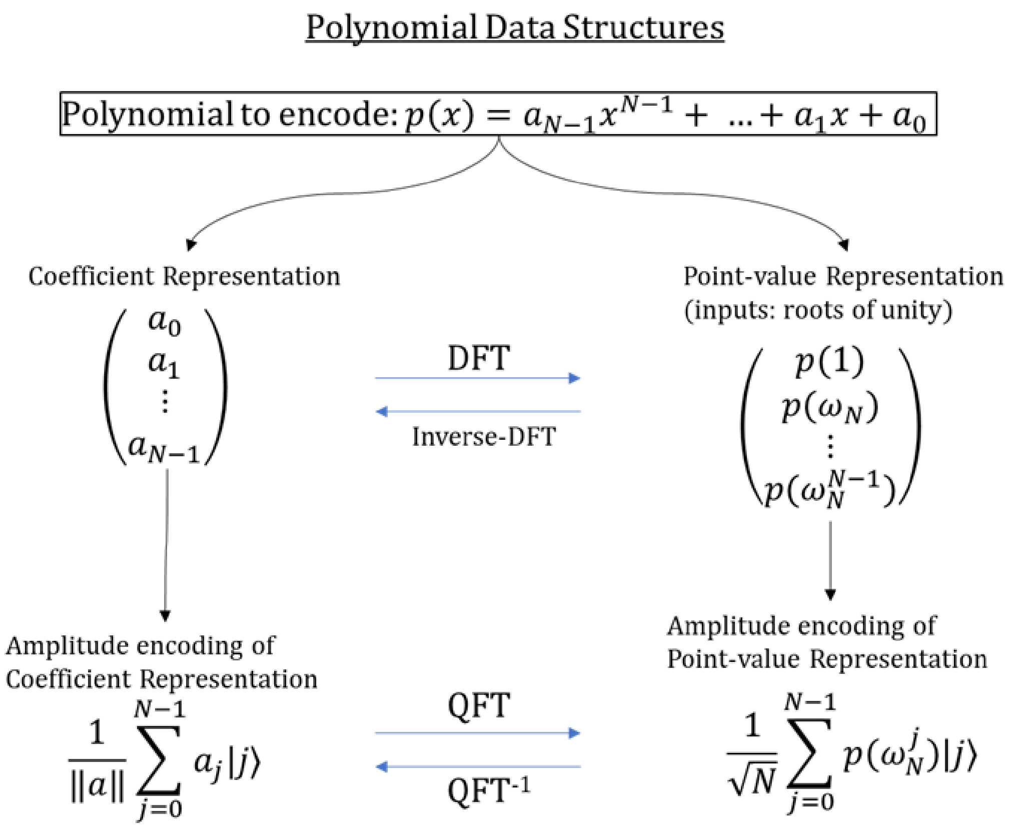
The second step of the QPE algorithm performs an inverse-QFT to convert from the amplitude encoding of the point-value representation to the amplitude encoding of the coefficient representation.

3 Quantum Phase Estimation

3.1 Setup

The Quantum Phase Estimation algorithm solves the following problem: Given a quantum state $|\psi\rangle$, and unitary U and such that $|\psi\rangle$ is an eigenvector of U, find the corresponding eigenvalue. $|\psi\rangle$ is given as one copy of a

Polynomial Data Structures



Polynomial Data Structures
Polynomial to encode: $p(x) = a_{N-1}x^{N-1} + \dots + a_1x + a_0$
Coefficient Representation
$\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \end{pmatrix}$
Point-value Representation (inputs: roots of unity)
$\begin{pmatrix} p(1) \\ p(\omega_N) \\ \vdots \\ p(\omega_N^{N-1}) \end{pmatrix}$
DFT
Inverse-DFT
Amplitude encoding of Coefficient Representation
$\frac{1}{\|a\|}\sum_{j=0}^{N-1} a_j|j\rangle$
Amplitude encoding of Point-value Representation
$\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} p(\omega_N^j)|j\rangle$
QFT
QFT^{-1}

Figure 1: This figure depicts the two polynomial data structures and their quantum analogues which are used in our explanation of QPE.

Approximating eigenvalues by roots of unity

Figure 2: The goal of QPE is to find the eigenvalue of U corresponding to $|\psi\rangle$. For a fixed precision N, QPE identifies the closest N-th root of unity to the true eigenvalue ($e^{i2\pi r}$). Thus we can see the goal of QPE as identifying the value k such that r is closest to $\frac{k}{N}$. Notice how increasing N will make the algorithm more accurate as there will be a finer discretisation of the circle.

quantum state, and the access to U is given as access to black-box calls to a generalised controlled-U gate.

We define the qudit-controlled-U^d gate, as the gate that acts in the following manner: $(C_d U^d)\,|j\rangle\,|\phi\rangle = |j\rangle\,U^j\,|\phi\rangle$. This is a generalisation of the standard controlled-U gate, which either applies the U gate 0 or 1 times, depending on the controlling register. The qudit-controlled-U^d gate applies the U gate j times, if the number in the controlling register is j.

The Quantum Phase Estimation algorithm is a quantum circuit that uses the qudit-controlled-U^d gates and the quantum state $|\psi\rangle$ to determine the eigenvalue.

Say $U\,|\psi\rangle = e^{i2\pi r}\,|\psi\rangle$. We can specify the bits of precision used in our algorithm as a parameter n (and define $N = 2^n$) such that $r \approx \frac{k}{N}$ where $k \in \mathbb{N}$. Let $\omega_N = e^{i2\pi/N}$, the principal N-th root of unity. Thus our goal is to determine k s.t. $U\,|\psi\rangle \approx \omega_N^k\,|\psi\rangle$ (See Figure 2).

3.2 Algorithm

This circuit has two parts. The first part **prepares an amplitude encoding of the point-value representation of the polynomial $\mathbf{x^k}$**. This can be shown from a simple derivation, notice that the resultant state has amplitudes equal to taking each root of unity to the k-th power:

$$\begin{aligned}
&C_d U^d[\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle\,|\psi\rangle] \\
&= \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}(\omega_N^k)^j\,|j\rangle\,|\psi\rangle \qquad\qquad (\text{Since } U\,|\psi\rangle = \omega_N^k\,|\psi\rangle) \\
&= \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}(\omega_N^j)^k\,|j\rangle\,|\psi\rangle
\end{aligned}$$

The second part **converts to the amplitude encoding of the coef-**

Quantum Phase Estimation Algorithm

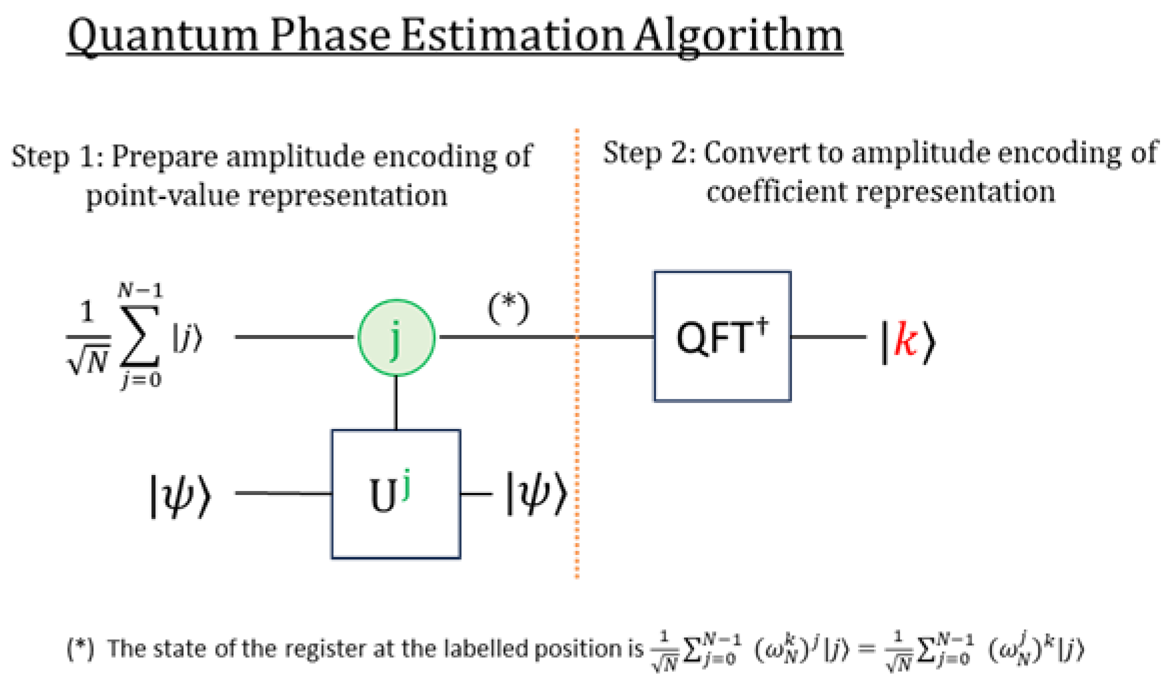

Quantum Phase Estimation Algorithm
Step 1: Prepare amplitude encoding of point-value representation
Step 2: Convert to amplitude encoding of coefficient representation
$\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle$
j
(*)
QFT†
$|k\rangle$
$|\psi\rangle$
U^j
$|\psi\rangle$
(*) The state of the register at the labelled position is $\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}(\omega_N^k)^j|j\rangle = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}(\omega_N^j)^k|j\rangle$

Figure 3: High level circuit explaining the Quantum Phase Estimation Algorithm. The algorithm has two steps. Step 1 prepares $\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}(\omega_N^k)^j|j\rangle = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}(\omega_N^j)^k|j\rangle$, which is the amplitude encoding of the point-value representation of the polynomial x^k. Step 2 converts to the amplitude encoding of the coefficient representation, yielding k.

ficient representation of the polynomial using the inverse-QFT, giving us the outcome $|k\rangle$ since the polynomial was x^k.

4 Other motivations

John Preskill and Watrous motivate it by starting from the Hadamard test.

I perceived the traditional explanations to leave the mechanism of the algorithm up to a computation. It would be left to a "computation by the reader", that the probability of measuring certain outcomes was 0 and hence any outcome is one of the desirable ones. This would mean that circuits were merely stated and could be checked to work by means of computation, but no motivation or explanation was given that might aid in memory or in coming up with new algorithms for oneself.

A story about QFT and AHSP can be told that sees QFT as mapping from group elements to characters

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Polynomials and the FFT*, ch. 30, pp. 898–925. MIT Press, 3rd ed., 2009.