Part III: Quantum Computation - Revision

Lectures by Richard Jozsa, notes by James Moore

1 Review of Shor's algorithm

1.1 Setting up Shor's algorithm

Definition: The *time complexity* of a quantum algorithm operating on n bits is the number of unitary gates used in the circuit. An algorithm has *polynomial time complexity* if the time complexity is polynomial in n.

Definition: Let N be an integer (with $n = O(\log(N))$ digits). The *factoring problem* is to determine a factor of N, not equal 1 or N, in polynomial time.

It was shown in the Part II course that the factoring problem is equivalent to the *periodicity-finding problem* via the Theorem:

Theorem: Choose a such that $1 < a < N$ with $\gcd(a, N) = 1$ (note we can check this in polynomial time). Then we can find a factor of N with probability strictly greater than $\frac{1}{2}$ by determining the period r of the function $f : \mathbb{Z} \to \mathbb{Z}_N$, given by $f(x) = a^x \pmod r$.

We thus switch to solving the periodicity-finding problem. Since we're working on a computer, we must work on a finite register $\mathbb{Z}_M$, so restrict the function f to $f : \mathbb{Z}_M \to \mathbb{Z}_N$ for M large.

ASIDE: Recall from Part II this creates technical issues from the fact that f is now no longer fully periodic. It was shown there that these issues are negligible provided $M = O(N^2)$, and thus in this course, we'll ignore those issues and assume f is fully periodic.

To implement f on a quantum computer, need to switch to using Hilbert spaces for output and input registers. Use $\mathcal{H}_M = \text{span}\{|i\rangle : i \in \mathbb{Z}_M\}$ and $\mathcal{H}_N = \text{span}\{|k\rangle : k \in \mathbb{Z}_N\}$, where $|i\rangle$ and $|k\rangle$ are orthonormal bases. Now use result:

Theorem: Any function $f : \mathbb{Z}_M \to \mathbb{Z}_N$ can be simulated by the operation $U_f : \mathcal{H}_M \otimes \mathcal{H}_N \to \mathcal{H}_M \otimes \mathcal{H}_N$ defined on a basis via

$$U_f |i\rangle |k\rangle = |i\rangle |k + f(i)\rangle ,$$

and extended to all of $\mathcal{H}_M \otimes \mathcal{H}_N$ by linearity. Furthermore, U_f is unitary, and if f can be computed in polynomial time, then so can U_f.

Proof: Obviously simulates f, since passing $|x\rangle |0\rangle$ to it gives $|x\rangle |f(x)\rangle$ back. U_f is unitary since it is a permutation of the basis vectors.

Finally, if f can be computed in polynomial time, just boost classical Boolean gates to quantum gates giving a polynomial complexity circuit for U_f. $\square$

Usually we suppose that we are given f in its quantum form U_f, with U_f's inner workings unknown to us.

Definition: If U_f's internal operation considered unknown, it is called a *blackbox* or *oracle*. The number of times we call U_f in an algorithm is called the *query complexity* of the algorithm.

We can now state the quantum form of the periodicity-finding problem:

Definition: Let U_f be a quantum oracle for the function $f : \mathbb{Z}_M \to \mathbb{Z}_N$, where f is efficiently computable (i.e. in polynomial time). Let $m = O(\log(M))$ be the number of digits of M. Suppose that:

- f is periodic, with unknown period $r \in \mathbb{Z}_N$. That is, $f(x + r) = f(x)$ for all $x \in \mathbb{Z}_M$, and r is the least such positive integer for which this occurs.
- f is one-to-one in each period (note this is guaranteed in the case of $f(x) = a^x$ from above, which we will use in Shor's algorithm). That is, for all $0 \leq x_1, x_2 < r$, we have $f(x_1) \neq f(x_2)$.

The *quantum period-finding problem* is to determine r in order $O(\text{poly}(m))$ time, with any prescribed success probability $1 - \epsilon$, for $\epsilon > 0$.

1.2 Shor's algorithm

Shor's Algorithm:

1. Make the state $\dfrac{1}{\sqrt{M}} \displaystyle\sum_{i=0}^{M-1} |i\rangle |0\rangle$. Query the oracle U_f with this state to get
$$\frac{1}{\sqrt{M}} \sum_{i=0}^{M-1} |i\rangle |f(i)\rangle .$$
2. Measure the second register to see $w = f(x_0)$, with $0 \leq x_0 < r$. Since f is periodic, it takes r distinct

values all with equal likelihood, so w is uniformly random, and hence x_0 is uniformly random on $0 \leq x_0 < r$.

By the Born rule, the first register collapses to

$$|\text{per}\rangle = \frac{1}{\sqrt{A}} \sum_{j=0}^{A-1} |x_0 + jr\rangle \,,$$

where $A = M/r$ is the number of periods. Discard the second register.

3. Apply the QFT to $|\text{per}\rangle$ (note this takes $O(m^2)$ time, where $m = O(\log(M))$): $QFT\,|\text{per}\rangle =$

$$\frac{1}{\sqrt{AM}} \sum_{y=0}^{M-1} \exp\left(\frac{2\pi i x_0 y}{M}\right) \left\{ \sum_{j=0}^{A-1} \exp\left(\frac{2\pi i j r y}{M}\right) \right\} |y\rangle \,.$$

Since $r/M = 1/A$, the sum in the curly brackets is

$$\sum_{j=0}^{A-1} \omega^{jy},$$

where ω is an Ath root of unity. Elementary algebra tells us this is non-zero, and equal to A, iff y is a multiple of A, say $y = kA$. Thus the transformed state can be written $QFT\,|\text{per}\rangle =$

$$\frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{2\pi i x_0 k A} |kA\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(\frac{2\pi i x_0 k M}{r}\right) |\frac{kM}{r}\rangle \,.$$

4. Measure $QFT\,|\text{per}\rangle$ to see some $c = k_0 M/r$, where k_0 is uniformly random in $0 \leq k_0 \leq r-1$. Rearranging we find:
$$\frac{c}{M} = \frac{k_0}{r}.$$
Both k_0 and r are unknown integers. If they were coprime, could cancel down c/M and read off the denominator to give r. If not, perform this procedure anyway and check: just compute $f(0)$ and $f(b)$ (where b is our proposed r) and compare.

5. If our proposed r is wrong, run the whole algorithm again repeatedly until we find the right r.

This algorithm works because of:

Theorem: The number of integers less than r and coprime to r grows as $O(r/\log(\log(r)))$. So in the above, Prob(k_0 coprime to r) $= O(1/\log(\log(r)))$.

Proof: Not required. □

Theorem: If a single trial of an experiment has success probability p, the probability of success in at least one of K trials is greater than $1 - \epsilon$, for any $0 < 1 - \epsilon < 1$, if $K \geq -\log(\epsilon)/p = O(1/p)$.

Proof: Elementary probability theory. □

Therefore, Shor's algorithm requires $K = O(\log(\log(r))) \leq O(\log(\log(M))) = O(\log(m))$ repeats to find r with any desired success probability. Hence it runs in polynomial time (as each step was polynomial - in particular, the Fourier transform needs $O(m^2)$ time).

1.3 A first look at shift-invariant states

The key to why Shor's algorithm works - and how it generalises - is *shift-invariant states*.

Definition: The map $k \mapsto k + x_0$ is called a *shift* by x_0. The corresponding quantum operation $U(x_0)$ is defined on a basis by $U(x_0)\,|k\rangle = |k + x_0\rangle$ and extended by linearity.

Theorem: The shift operators $U(x_0)$ obey (i) $U(x_0)$ is unitary for all x_0; (ii) $U(x+y) = U(x)U(y)$ for all x, y; (iii) $\{U(x)\}_{x \in \mathbb{Z}_M}$ are simultaneously diagonalisable.

Proof: (i) Clear since permutation. (ii) Clear from definition. (iii) All $U(x)$'s commute by (ii). □

Definition: An element of the simultaneous, orthonormal basis of common eigenvectors of $\{U(x)\}$, written $\{|\chi_k\rangle\}_{k \in \mathbb{Z}_M}$, is called a *shift invariant state*. We write $U(x_0) = \omega(x_0, k)\,|\chi_k\rangle$ where $\omega(x_0, k)$ is some eigenvalue. Since U is unitary, $|\omega(x_0, k)| = 1$.

Shift-invariant states inspire the above algorithm as follows. Define
$$|S\rangle = \frac{1}{\sqrt{|S|}} \sum_{s \in S} |s\rangle \,,$$
for any set S. If $R = \{0, r, 2r, ..., (A-1)r\}$, then we have can write our periodic state as $|\text{per}\rangle = |x_0 + R\rangle = U(x_0)\,|R\rangle$ in terms of the shift operator.

Write $|R\rangle$ in the shift-invariant basis: $|R\rangle = \sum_{k=0}^{M-1} a_k\,|\chi_r\rangle$.
Therefore:
$$|\text{per}\rangle = \sum_{k=0}^{M-1} a_k \omega(x_0, k)\,|\chi_k\rangle \,.$$

Note that the a_k's depend on r, but x_0 and k (which is a dummy variable) do not. So measuring $|\text{per}\rangle$ with respect to the shift-invariant basis must give information about r!

The probability of getting χ_k is $|a_k \omega(x_0, k)|^2 = |a_k|^2$, which is purely dependent on r and not x_0. This is the secret of Shor's algorithm!

How do we measure in such a basis $|\chi_k\rangle$ then? Well, we rotate $|\chi_k\rangle$ back to the standard basis first, and then perform the measurement. This will be our generalised definition of the quantum Fourier transform:

Definition: The *quantum Fourier transform* is the map defined by $QFT\,|\chi_k\rangle = |k\rangle$, and extended by linearity.

This is unitary because it maps an orthonormal basis to an orthonormal basis.

Thus applying the QFT as defined here, and then performing a standard basis measurement, implements the $|\chi_k\rangle$ basis measurement.

It's all very well *defining* an operation like this, but can we actually do it? We need an explicit formula for it, one that agrees with our original QFT definition hopefully!

Theorem: An explicit formula for $|\chi_k\rangle$ is:

$$|\chi_k\rangle = \frac{1}{\sqrt{M}} \sum_{l=0}^{M-1} e^{-2\pi i k l/M} \,|l\rangle \,.$$

Proof: Just need to verify this. We have $U(x_0)\,|\chi_k\rangle =$

$$\frac{1}{\sqrt{m}} \sum_{l=0}^{M-1} e^{-2\pi i k l/M} \,|l + x_0\rangle = \frac{1}{\sqrt{M}} \sum_{\tilde{l}=0}^{M-1} e^{2\pi i k(\tilde{l}-x_0)/M} \,|\tilde{l}\rangle \,,$$

on substituting $\tilde{l} = l + x_0$. Factoring out a phase, we have $U(x_0)\,|\chi_k\rangle = e^{2\pi i k x_0/M}\,|\chi_k\rangle$. □

Theorem: The QFT, as defined in terms of the shift invariant states, has matrix elements:

$$[QFT]_{kl} = \frac{1}{\sqrt{M}} e^{2\pi i l k/M},$$

as expected.

Proof: We have $QFT\,|\chi_k\rangle = |k\rangle \Rightarrow |\chi_k\rangle = QFT^{-1}\,|k\rangle$. Therefore we can read off from above:

$$[QFT^{-1}]_{lk} = \frac{1}{\sqrt{M}} e^{-2\pi i l k/M}.$$

Now take conjugate transpose, since QFT is unitary. □

---

2 The hidden subgroup problem

2.1 The hidden subgroup problem

Definition: Let G be a group and let $K \leq G$ be a subgroup. Let $f : G \to X$ is a function implemented by the quantum oracle U_f. Suppose we are promised that f is constant on the left cosets of K in G, and f is distinct on distinct cosets. The *hidden subgroup problem* is to 'determine' the subgroup K in time $O(\text{poly}(\log(|G|)))$, with any constant probability $1 - \epsilon < 1$.

The word 'determine' is a little fuzzy here, as there are many ways to present a subgroup. Examples of solutions might include outputting a set of generators of K, or giving a uniform sample of elements from K.

Note the time complexity we want to complete the problem in is exponentially faster than just checking all group elements.

Example 1: The period-finding problem is a hidden subgroup problem. The group is $G = \mathbb{Z}_M$, the function is $f : \mathbb{Z}_M \to X$, and the hidden subgroup is $K = \{0, r, 2r, ..., (A-1)r\} \leq G$. Clearly f is constant on cosets.

To present K, we determine its generator, r.

Example 2: Let $p \in \mathbb{Z}$ be prime and let $\mathbb{Z}_p^* = \{1, 2, .., p-1\}$ be the multiplicative group modulo p. An element $g \in \mathbb{Z}_p$ is a *primitive root* if powers of g generate $\mathbb{Z}_p^*$. It's a fact from number theory that primitive roots always exist for prime p.

Let g be a primitive root. Then $x \in \mathbb{Z}_p^*$ may be written $x = g^y$, where $y \in \{0, ..., p-2\} \in \mathbb{Z}_{p-1}$. The power $y = \log_g(x)$ is called the *discrete logarithm* of x to the base g. The *discrete logarithm problem* is to determine $\log_g(x)$ given g and x.

The discrete logarithm problem is a HSP. Consider $f : \mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1} \to \mathbb{Z}_p^*$ given by $f(a, b) = g^a x^{-b} (\text{mod } p)$. This is the same as $f(a, b) = g^{a-yb}$ where y is the discrete logarithm.

Notice that $f(a, b) = c$, where $c = g^k$ is some fixed value, implies $g^{a-yb} = g^k$, and so $a - yb - k \equiv 0 \pmod{p-1}$. Parametrise the solution as $b = \lambda$, $a = b\lambda + k$ for some $\lambda \in \mathbb{Z}_{p-1}$. Then $f(a, b) = c$ if and only if $(a, b) = \lambda(y, 1) + (k, 0)$.

Thus the hidden subgroup is the group generated by $(y, 1)$, and the above result proves that f is constant and distinct on the cosets of $\langle (y, 1) \rangle$. Finding the subgroup is equivalent to solving the discrete logarithm problem.

2.2 Shift-invariant states for Abelian G

To solve the Abelian HSP, we use the shift-invariant states technique as in Shor's algorithm. We restrict to Abelian groups G now, and will discuss non-Abelian groups later. To construct the states, we use the following approach.

Definition: An *irreducible representation* of the group G on $\mathbb{C}^* = (\mathbb{C}\backslash\{0\}, \times)$ is a mapping $\chi : G \to \mathbb{C}^*$ satisfying $\chi(g_1 + g_2) = \chi(g_1)\chi(g_2)$, i.e. it is a group homomorphism.

Theorem: We have the following properties:

(i) $\chi(g)$ is a $|G|$th root of unity; hence $\chi : G \to S^1$, the unit circle.

(ii) SCHUR'S LEMMA: If χ_i and χ_j are both irreps, then

$$\sum_{g\in G} \chi_i(g)\overline{\chi_j(g)} = \delta_{ij}|G|.$$

(iii) There are exactly $|G|$ different irreps and no more.

Proof: (i) Since G is finite, there exists r (the *order* of g) such that $g + g + ... + g$ (r times) $= 0$. Hence $1 = \chi(0) = \chi(g + ... + g) = \chi(g)^r$. Recall r divides $|G|$ by Lagrange's Theorem, and so $\chi(g)^r = 1 \Rightarrow \chi(g)^{|G|} = 1$.

(ii) Define $S = \sum_{g\in G} \chi_i(g)\overline{\chi_j(g)}$. Note that

$$\chi_i(h)S = \sum_{g\in G} \chi_i(g+h)\overline{\chi_j(g)} = \sum_{\tilde{g}\in G} \chi_i(\tilde{g})\overline{\chi_j(\tilde{g}-h)},$$

where $\tilde{g} = h + g$. Now using the homomorphism property, and $\overline{\chi_j(-h)} = \chi_j(h)$ (since inverses preserved by homomorphism, and $\chi_j(h)$ is unit modulus), we have

$$\chi_i(h)S = \chi_j(h)S \Rightarrow (\chi_i(h) - \chi_j(h))S = 0.$$

If $\chi_i \neq \chi_j$, there exists h where they disagree. Hence $S = 0$ if $i \neq j$. If $i = j$, then from (i) the sum is trivially $|G|$.

(iii) Not required in this course. □

By (iii) we can label the irreps as χ_g for $g \in G$.

Example: $\chi(g) = 1$ for all g is an irrep, trivially. This is called the *trivial representation* and we label it as χ_0.

Theorem: For any $\chi \neq \chi_0$, we have $\sum_{g\in G} \chi(g) = 0$.

Proof: Use Schur's Lemma with $\chi_j = \chi_0$ and $\chi_i \neq \chi_0$. □

Define the state space $\mathcal{H}_{|G|}$ to be the span of the orthonormal basis $\{|g\rangle\}_{g\in G}$. Then, as in Shor's algorithm, we define:

Definition: The *shift operators* $U(k)$ are defined by $U(k)|g\rangle = |g + k\rangle$, and extended by linearity (note $+$ is the group operation here).

Theorem: We have: (i) $U(k)$ is unitary; (ii) $U(h)U(k) = U(k)U(h)$; (iii) $\{U(h)\}_{h\in G}$ are simultaneously diagonalisable.

Proof: (i) Just a permutation. (ii) Since group Abelian. (iii) Immediate from (ii). □

Definition: The common orthonormal eigenbasis of $\{U(k)\}_{k\in G}$ is called the *shift-invariant basis*, written $\{|\chi_k\rangle\}_{k\in G}$, and its elements are *shift-invariant states*.

Definition: The *quantum Fourier transform* on the group G is the unitary operation QFT satisfying $QFT\,|\chi_k\rangle = |k\rangle$, and extended by linearity.

As before, we can explicitly construct the shift-invariant states and the QFT.

Theorem: The shift-invariant states are given by

$$|\chi_k\rangle = \frac{1}{\sqrt{|G|}}\sum_{g\in G} \overline{\chi_k(g)}\,|g\rangle\,.$$

The states are indeed orthonormal, as claimed.

Proof: We have $U(g)\,|\chi_k\rangle =$

$$\frac{1}{\sqrt{|G|}}\sum_{h\in G} \overline{\chi_k(h)}\,|h+g\rangle = \frac{1}{\sqrt{|G|}}\sum_{h'\in G} \overline{\chi_k(h'-g)}\,|h'\rangle\,,$$

where we've let $h' = h + g$. Using the properties of the irreps, we can factor out $\chi_k(g)$ leaving $U(g)\,|\chi_k\rangle = \chi_g(g)\,|\chi_k\rangle$. So indeed these are shift-invariant states.

To prove orthonormality, use Schur's Lemma. □

Theorem: The QFT has explicit matrix representation

$$[QFT]_{kg} = \frac{1}{\sqrt{|G|}}\chi_k(g),$$

so that $QFT\,|g\rangle = \frac{1}{\sqrt{|G|}}\sum_{k\in G} \chi_k(g)\,|k\rangle$.

Proof: By definition, $|\chi_k\rangle = QFT^{-1}\,|k\rangle$. So reading off the matrix elements, we see

$$[QFT^{-1}]_{gk} = \frac{1}{\sqrt{|G|}}\overline{\chi_k(g)}.$$

Since QFT is unitary, take conjugate transpose. □

Example 1: For $G = (\mathbb{Z}_M, +)$ we can check that $\chi_a(b) = e^{2\pi iab/M}$ are indeed irreps for each a.

Example 2: For $G = \mathbb{Z}_{M_1} \times ... \times \mathbb{Z}_{M_r}$, if $g_1 = (a_1, ..., a_r)$ and $g_2 = (b_1, ..., b_r)$, then

$$\chi_{g_1}(g_2) = \exp\left(2\pi i\left(\frac{a_1b_1}{M_1} + ... + \frac{a_rb_r}{M_r}\right)\right)$$

are irreps for each g_1. In particular, because the exponential factors, we have that $QFT_G = QFT_{M_1} \otimes ... \otimes QFT_{M_r}$.

In fact, Example 2 exhausts all possible Abelian groups because:

Theorem: Any finite Abelian group is isomorphic to a direct product of the form $\mathbb{Z}_{M_1} \times ... \times \mathbb{Z}_{M_r}$, where the M_i are prime powers, $p_1^{s_1}$, ... , $p_r^{s_r}$.

Proof: Not required. □

2.3 The Abelian HSP algorithm

The algorithm for the Abelian HSP is the same as Shor's algorithm, but instead using the Fourier transform on the group G as defined in terms of shift-invariant states.

HSP Algorithm: Consider $f : G \to X$ with hidden subgroup $K \leq G$, and f constant and distinct on cosets of K. Let $|G|/|K| = m$. Consider states with basis $\{|g\rangle, |x\rangle\}_{g\in G, x\in X}$.

1. Make the states $\frac{1}{\sqrt{|G|}}\sum_{g\in G}|g\rangle|0\rangle$, and query the oracle U_f with it to get
$$\frac{1}{\sqrt{|G|}}\sum_{g\in G}|g\rangle|f(g)\rangle .$$
2. Measure the second register to see some $f(g_0)$ for $g_0 \in G$ a uniformly random representative of one of the cosets of K. Then the remaining first register gives the coset state:
$$|g_0 + K\rangle = \frac{1}{\sqrt{|K|}}\sum_{k\in K}|g_0 + k\rangle = U(g_0)|K\rangle .$$
Discard the second register.
3. Apply the group quantum Fourier transform to get:
$$QFT|g_0 + K\rangle = \frac{1}{\sqrt{|G|}}\frac{1}{\sqrt{|K|}}\sum_{l\in G}\chi_l(g_0)\left\{\sum_{k\in K}\chi_l(k)\right\}|l\rangle .$$
Recall the term in braces is zero if $\chi_l \neq \chi_0$, the trivial rep. BUT these irreps are summing only over K here, so it's sufficient that χ_l *restricts* to χ_0 on K for it to be non-zero. If χ_l restricts to χ_0 on K, then the sum is equal to $|K|$ trivially. Thus we have
$$QFT|g_0 + K\rangle = \sqrt{\frac{|K|}{|G|}}\sum_{\substack{l\in G \\ \chi_l = \chi_0 \text{ on } K}}\chi_l(g_0)|l\rangle .$$
4. Measure. Since $|\chi_l(g_0)|^2 = 1$, the measurement is independent of g_0. We see a uniformly random l such that $\chi_l(k) = 1$ for all $k \in K$.

How do we now extract information about K? We have the following:

Theorem: For any subgroup $K \leq G$ (even for G non-Abelian), K has a set generators k_1, k_2, ... ,k_m, where $m = O(\log(|K|)) = O(\log(|G|))$.

Proof: Not required. □

Theorem: Suppose we repeat the algorithm N times, so that we have equations $\chi_{l_1}(k) = 1$, $\chi_{l_2}(k) = 1$, ... , $\chi_{l_N}(k) = 1$. Then provided $N = O(\log(|G|))$, the equations suffice to determine a generating set of K with any good probability.

Proof: Not required. □

This result shows that the algorithm indeed gives us a solution to the Abelian HSP.

Example: Let $G = \mathbb{Z}_{M_1} \times \mathbb{Z}_{M_2} \times ... \times \mathbb{Z}_{M_q}$. Recall for $l = (l_1, ..., l_q)$ and $g = (b_1, ..., b_q)$ we have
$$\chi_l(g) = \exp\left(2\pi i\left(\frac{l_1 b_1}{M_1} + ... + \frac{l_q b_q}{M_q}\right)\right).$$
Hence writing $k = (k_1, ..., k_q)$ the equation $\chi_l(k) = 1$ is
$$\frac{l_1 b_1}{M_1} + ... + \frac{l_q b_q}{M_q} = 0 \text{ (mod } 1).$$
(Modulo 1 in this context means the LHS is an integer.) This is a homogeneous linear equation in k, and we want the kernel. Provided we have $O(\log(|K|))$ such equations, we can determine the null space K as the null space.

2.4 Example: generalised Simon's problem

Definition: The *generalised Simon's problem* is defined as follows. Let U_f be an oracle for $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ which is 2^k to 1, with k linearly independent n-bit strings a_1, a_2, ... , $a_k \in \mathbb{Z}_2^n$ that obey $f(x) = f(x \oplus a_i)$ for all $x \in \mathbb{Z}_2^n$ (here, $\oplus$ is bitwise addition). The problem is to output any $b \in \mathbb{Z}_n^2$ having $f(x) = f(x \oplus b)$ for all $x \in \mathbb{Z}_2^n$ in polynomial time.

Theorem: Simon's problem is an Abelian HSP.

Proof: We claim $K = \langle a_1, ..., a_k\rangle$, the subgroup generated by the a_i is the hidden subgroup. It's a subgroup by definition. The problem is Abelian since $\mathbb{Z}_2^n$ is Abelian.

The cosets of K are $x \oplus \langle a_1, a_2, ..., a_k\rangle$ so a general element is $x \oplus a_1^{r_1} \oplus ... \oplus a_k^{r_k}$, $r_i = 0, 1$. Note that
$$f(x \oplus a_1^{r_1} \oplus ... \oplus a_k^{r_k}) = f(x),$$
so indeed constant on cosets.

Finally, need f distinct on cosets. Each coset is of size 2^k since the a_i are linearly independent. f is given as 2^k to 1, and hence f must be distinct on each coset. □

Theorem: The irreps for this group are $\chi_x(y) = (-1)^{x \cdot y}$, where $\cdot$ is the bitwise inner product, and the shift invariant states are:

$$|\chi_x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \mathbb{Z}_2^n} (-1)^{x \cdot y} |y\rangle .$$

The Fourier transform on the group is given by:

$$QFT = \underbrace{H \otimes H \otimes ... \otimes H}_{n \text{ times}} = H^n,$$

where H is the Hadamard gate.

Proof: From the general theory, the irrep $\chi_x(y)$ is given by:

$$\chi_x(y) = \exp\left(2\pi i \left(\frac{x_1 y_1}{2} + \frac{x_2 y_2}{2} + ... + \frac{x_n y_n}{2}\right)\right) = (-1)^{x \cdot y}.$$

The shift-invariant states then come from the general theory. The Fourier transform is, from the general formula,

$$QFT\,|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \mathbb{Z}_2^n} (-1)^{x \cdot y} |y\rangle .$$

Notice that $H^n |x\rangle$ is the uniform superposition, except we pick up a minus sign whenever there's a one in x; this can be accounted for by including a factor of $(-1)^{x \cdot y}$ next to $|y\rangle$ in the expansion. □

Example: Let's see how the HSP algorithm works in this case.

1. Make the state $\frac{1}{\sqrt{2^n}} \sum_{x \in \mathbb{Z}_2^n} |y\rangle |0\rangle$, and apply U_f to get
$$\frac{1}{\sqrt{2^n}} \sum_{x \in \mathbb{Z}_2^n} |x\rangle |f(x)\rangle .$$
2. Measure the second register and then discard it. We see some $f(x_0)$ and the state collapses to the coset state
$$\frac{1}{\sqrt{2^k}} \sum_{x \in K} |x_0 + x\rangle ,$$
where $2^k = |K|$ (recall from above).
3. Now apply the group's QFT as derived above. We get the state:
$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \frac{1}{\sqrt{2^k}} \sum_{y \in \mathbb{Z}_2^n} (-1)^{x_0 \cdot y} \left\{ \sum_{x \in K} (-1)^{x \cdot y} \right\} |y\rangle .$$

As usual, we now consider the sum in curly braces. We could use the general theory here, and require the rep to restrict the trivial rep on K; this is the condition that the sum is zero unless $(-1)^{x \cdot y} = 1$ for all $x \in K$, i.e. $x \cdot y \equiv 0 \pmod 2$ for all $x \in K$.

However, it's also possible to show this by elementary means:

Theorem: Either $x \cdot y \equiv 0 \pmod 2$ for all $x \in K$, or $x \cdot y \equiv 0 \pmod 2$ for exactly half of $x \in K$.

Proof: If $x \cdot y \equiv 0$ for all $x \in K$, we're done. So suppose $x_1 \cdot y \equiv 1$ and for all $x' \notin \text{span}\{x_1\}$, we have $x' \cdot y \equiv 0$. Then for any $x_2 \notin \text{span}\{x_1\}$, we have $0 \equiv (x_1 + x_2) \cdot y \equiv x_1 \cdot y + x_2 \cdot y \equiv 1$. This is a contradiction. Therefore, there exists $x_2 \notin \text{span}\{x_1\}$ such that $x_1 \cdot y \equiv 1$.

Now suppose for all $x' \notin \text{span}\{x_1, x_2\}$, we have $x' \cdot y \equiv 0$. Then for any x_3 $/\!\int$ $\text{span}\{x_1, x_2\}$, we have $0 \equiv (x_1 + x_3) \cdot y \equiv x_1 \cdot y + x_3 \cdot y \equiv 1$. Again, a contradiction.

Iterate until we have built up $\text{span}\{x_1, x_2, ..., x_k\}$ where $x_i \cdot y \equiv 1$ for all x_i. Then this spans all of K since it has size 2^k. Thus for any $x \in K$, we have $x = x_1^{r_1} \oplus ... \oplus x_k^{r_k}$, where $r_i = 0, 1$. Thus:

$$x \cdot y \equiv \begin{cases} 0 & \text{even number of } r_i \text{ equal to } 1 \\ 1 & \text{odd number of } r_i \text{ equal to } 1. \end{cases}$$

So half/half split in K as expected. □

Using this fact, we have

$$\sum_{x \in K} (-1)^{x \cdot y} = \begin{cases} 0 & \text{if } y \text{ is not orthogonal to } K \\ |K| & \text{if } y \text{ is orthogonal to } K. \end{cases}$$

Hence $|\psi\rangle = \sqrt{\frac{2^k}{2^n}} \sum_{\substack{y \in \mathbb{Z}_2^n \\ y \perp K}} (-1)^{x_0 \cdot y} |y\rangle$. Back to algorithm...

4. Now measure. We see some y orthogonal to K, uniformly random in the orthogonal complement of K.
5. Run the procedure $n - k$ times to see y_1, ... , y_{n-k}, all orthogonal to K. We now need to examine how likely it is for these strings to span the orthogonal complement of K (for then we just pick any b *not* in the span of the y_i to solve Simon's problem).

To examine the likelihood that these strings span, we use:

Theorem: If we pick m m-bit strings, y_1, ... , y_m uniformly at random from $\mathbb{Z}_2^n$, then they will be linearly independent, and not include the all-zero string, with probability at least $1/4$.

Proof: Pick the strings one by one. Only restriction on y_1 is that it is not zero. So chance of good y_1 is $1 - 1/2^m$.

y_2 can't be in the subspace spanned by y_1, which has 2 elements. Thus chance of good y_2 is $1 - 1/2^m$.

Continuing in this fashion, we see the probability of selecting m good strings is:

$$\prod_{j=1}^{m}\left(1-\frac{2^{j-1}}{2^m}\right).$$

Then using the inequality $(1-a)(1-b) \geq 1-(a+b)$ for $a, b \in [0,1]$ repeatedly, we find:

$$\prod_{j=1}^{m}\left(1-\frac{2^{j-1}}{2^m}\right) = \frac{1}{2}\sum_{j=1}^{m-1}\left(1-\frac{2^{j-1}}{2^m}\right) \geq \frac{1}{2}\left(1-\sum_{j=1}^{m-1}\frac{2^{j-1}}{2^m}\right)$$
$$= \frac{1}{2}\left(1-\left(\frac{2^{m-1}-1}{2^m}\right)\right) = \frac{1}{4}+\frac{1}{2^{m+1}} \geq \frac{1}{4}. \quad \square$$

Now go back to our observed strings y_1, y_2, ... , y_{n-k}. Each has n bits, but consider only the first $n-k$ in each case. The chance that the first $n-k$ bits of all the strings are all linearly independent is thus $\geq 1/4$ by the above.

So span$\{y_1, ..., y_{n-k}\}$ is indeed the orthogonal complement of K with likelihood $\geq 1/4$, and so we solve Simon's problem with probability $\geq 1/4$.

By the Probability Lemma, this is sufficient to run the algorithm in polynomial time if we want to solve it with any high probability. To spell it out, if we fail to get a solution of Simon's problem, just repeat the whole algorithm. The probability of success on the Cth trial is

$$\left(\frac{3}{4}\right)^{C-1}\cdot\frac{1}{4}.$$

So for this to be $> 1-\epsilon$ for any ϵ, we just need $C > 1+\log(4(1-\epsilon))/\log(3/4)$. So we only need a constant number of repeats, with $n-k$ queries to oracle at each repeat. Thus algorithm is of order $O(n)$.

2.5 Non-Abelian HSP

Write G multiplicatively now and no longer assume G is Abelian. The first few steps of the algorithm can be run as before:

1. Make $\frac{1}{\sqrt{|G|}}\sum_{g\in G}|g\rangle|0\rangle$, and apply U_f to get:
$$\frac{1}{\sqrt{|G|}}\sum_{g\in G}|g\rangle|f(g)\rangle.$$
2. Measure the second register to see some $f(g_0)$. Discard it to leave the coset state
$$|g_0K\rangle = \frac{1}{\sqrt{|K|}}\sum_{k\in K}|g_0k\rangle.$$

Normally, the algorithm tells us to apply the QFT here. But problems arise, as it turns out there is no shift-invariant basis (the U's don't commute, so don't have a common eigenbasis)! *However*, we still *can* construct the QFT...

The Non-Abelian Quantum Fourier Transform: Begin by generalising the notion of an irrep (c.f. Symmetries, Fields and Particles):

Definition: A *d-dimensional representation* of a group G is a group homomorphism $\chi : G \to U(d)$ where $U(d)$ is the group of $d\times d$ unitary matrices.

A representation is called *irreducible* if no subspace of $\mathbb{C}^d$ is left invariant by all the matrices in the representation, $\chi(g)$, $g \in G$.

That is, a representation is irreducible if you can't block diagonalise all the matrices in the representation on some block simultaneously by some basis change.

A *complete set* of irreps is a set χ_1, χ_2, ... , χ_m of irreps such that any irrep is unitarily equivalent to one of the χ_i (here, equivalent means 'the same up to a change of basis', $\chi' = V\chi V^{-1}$, $V \in U(d)$).

Theorem: Let χ_1, χ_2, ... , χ_m be a complete set of irreps of a group G with dimensions d_1, d_2, ... , d_m. Then:

1. $d_1^2 + d_2^2 + ... + d_m^2 = |G|$;
2. Schur's Lemma: Let $\chi_{i,jk}(g)$ be the (j,k)th entry of the matrix $\chi_i(g)$, for $j, k = 1, ..., d_i$. Then
$$\sum_{g\in G}\chi_{i,jk}(g)\overline{\chi_{i',j'k'}(g)} = |G|\delta_{ii'}\delta_{jj'}\delta_{kk'}.$$

Proof: Not required. □

It follows from the above that

$$|\chi_{i,jk}\rangle = \frac{1}{\sqrt{|G|}}\sum_{g\in G}\overline{\chi_{i,jk}(g)}\,|g\rangle$$

are an orthonormal basis. So there exists a unitary operation transforming $\{|\chi_{i,jk}\rangle\}$ into $\{|g\rangle\}$.

Definition: The *quantum Fourier transform* is the unitary operation defined by $QFT\,|\chi_{i,jk}\rangle = |g\rangle$, and extended by linearity.

However this does not provide us with the algorithm we want. The $|\chi_{i,jk}\rangle$ are *not* shift-invariant states. In particular, a measurement of the coset state $|g_0K\rangle$ in the above basis does not give an output distribution that is independent of g_0.

On the other hand, a 'partial' shift invariant survives. Consider the *incomplete* measurement M_{rep} on $|g_0K\rangle$ that distinguishes only the irreps (i.e. the i values) and not all i, j, k's. That is, the measurement has outcome i associated to the d_i^2-dimensional orthogonal subspace spanned by $\{|\chi_{i,jk}\rangle : j, k \in 1...d_i\}$.

Then since $\chi_i(g_1 g_2) = \chi_i(g_1)\chi_i(g_2)$, we can disentangle $\chi_i(g_0)$ on measurement, giving us an outcome distribution which *is* independent of g_0.

This gives direct, but *incomplete*, information about K. For example, it's possible to show that conjugate subgroups such as K and $L = g'Kg'^{-1}$, give the same output distribution.

We also need (for an efficient HSP algorithm) the QFT to run in $O(\text{poly}(\log(|G|)))$ time. This is true for any Abelian G, and some non-Abelian groups, but not all. Even for the ones non-Abelian ones where we do know how to implement the QFT efficiently, there's still no efficient HSP algorithm!

Int the Abelian case, QFT implementation is achieved using the *fast Fourier transform*. This technique is also applicable to non-Abelian permutation groups.

KNOWN RESULTS:
Some partial results on the non-Abelian HSP are:

Theorem (Hallgren and Russell): If G has an efficient QFT, and K is given to be a normal subgroup of G, then there is an efficient HSP quantum algorithm.

Theorem (Ettinger, Høyer and Knill): For a general non-Abelian HSP, $M = O(\text{poly}(\log(|G|)))$ random coset states $\{|g_1 K\rangle, ..., |g_M K\rangle\}$ suffice to determine K.

The problem with the Ettinger, Høyer and Knill Theorem is that it is not known how to *efficiently* determine K from those cosets!

2.6 Non-Abelian example: graph problems

Definition: A *graph* is a set of edges and vertices, $A = \{\text{vertices } V, \text{edges } E\}$. We assume our graphs are *undirected* and there is at most one edge between any two vertices. Label the vertices by $[n] = \{1, 2, ..., n\}$.

Notation: The *permutation group* on the vertices is written $\mathcal{P}_n$.

Definition: The *automorphism group* of a graph A, denoted Aut(A) is a subgroup of $\mathcal{P}_n$ containing the permutation $\pi \in \mathcal{P}_n$ such that $i - j$ is an edge in A iff $\pi(i) - \pi(j)$ is an edge in A.

That is, if the labelled graph $\pi(A)$ (i.e. replace label i in A by $\pi(i)$) is the *same* as the labelled graph A.

Definition: The *graph automorphism HSP* is defined as follows. Let $G = \mathcal{P}_n$ and let X be the set of all labelled graphs on n vertices. Define

$$f_A(\pi) = \pi(A).$$

The problem is to determine the hidden subgroup $K = \text{Aut}(A)$.

Indeed, this is a HSP, since K is obviously a subgroup of $\mathcal{P}_n$, and a general coset element is of the form $\rho\pi \in \rho\text{Aut}(A)$ so that

$$f_A(\rho\pi) = \rho\pi(A) = \rho(A).$$

So constant on cosets.

This (non-Abelian) HSP has applications to the *graph isomorphism problem*.

Definition: Two labelled graphs A and B (with n vertices) are isomorphic if there exists $\pi \in \mathcal{P}_n$ such that $\pi(A) = B$. That is, $i - j$ is an edge in A iff $\pi(i) - \pi(j)$ is an edge in B.

Definition: The *graph isomorphism problem* is to determine whether two graphs A and B are isomorphic.

Theorem: We can reduce the graph isomorphism problem to the graph automorphism HSP.

Proof: Let C be the graph on $2n$ vertices which is the disjoint union of A and B. Label the vertices using $[2n] = \{1, ..., 2n\}$, where $L_A = \{1, 2, ..., n\}$ label A and $L_B = \{n+1, ..., 2n\}$ label B.

Consider Aut$(C) \leq \mathcal{P}_{2n}$. We have $\pi \in \text{Aut}(C)$ if $i - j$ is an edge in C iff $\pi(i) - \pi(j)$ is an edge in C. By induction, this is the same as the condition: $i - k_1 - ... - k_l - j$ is a path in C iff $\pi(i) - \pi(k_1) - ... - \pi(k_l) - \pi(j)$ is a path in C.

Recall C is disjoint. So if $i, j \in L_A$ and $\pi(i) \in L_A$, $\pi(j) \in L_B$, then there is a path from i to j in C, but no path from $\pi(i)$ to $\pi(j)$ in C. So $\pi \notin \text{Aut}(C)$. Thus if $\pi \in \text{Aut}(C)$, it must either swap L_A and L_B fully, or permute L_A and L_B separately into themselves.

Let $H = \{\pi \in \text{Aut}(C) : \pi \text{ permutes } L_A, L_B \text{ separately}\}$. Then if A and B are not isomorphic, Aut$(C) = H$, clearly. If A and B are isomorphic, there exists μ such that μ swaps L_A and L_B fully. We claim Aut$(C) = H \cup \mu H$.

To prove this simply note that if $\pi \in \text{Aut}(C)$, it is either in H, or swaps L_A and L_B fully. In the latter case, $\mu^{-1}\pi \in H$, since μ^{-1} swaps everything back again! Therefore, $\pi \in \mu H$. Thus Aut$(C) = H \cup \mu H$, as required.

So if the graphs are isomorphic, $|H| = |\mu H| = \frac{1}{2}|\text{Aut}(C)|$, and if not, $|H| = |\text{Aut}(C)|$. Supposing we have a HSP algorithm for the graph automorphism problem, we can apply this to $\text{Aut}(C)$ to determine a, say, random sample from it. Trialling elements from the random sample we can determine which regime we are in, solving graph isomorphism. □

We now present a couple of ideas for graph isomorphism/automorphism algorithms that don't work (but are hopefully give insight as to why).

Example (The swap test): The *swap test* is defined as follows. Given states $|\alpha\rangle, |\beta\rangle \in \mathcal{H}_d$, adjoint an extra qubit to start with $|0\rangle|\alpha\rangle|\beta\rangle \in \mathcal{H}_2 \otimes \mathcal{H}_d \otimes \mathcal{H}_d$.

Now apply the followings actions: (i) apply H to the qubit; (ii) apply the controlled SWAP gate, controlled by the qubit (note $\text{SWAP}|0\rangle|\alpha\rangle|\beta\rangle = |0\rangle|\alpha\rangle|\beta\rangle$ and $\text{SWAP}|1\rangle|\alpha\rangle|\beta\rangle = |1\rangle|\beta\rangle|\alpha\rangle$); (iii) apply H to the qubit again; (iv) measure the qubit. If we see 0 output 'near' and if we see 1, output 'far'.

What's the point of this test? It's easy to show that the probability of 0 is

$$\frac{1 + |\langle\alpha|\beta\rangle|^2}{2},$$

so the result depends on the angle between $|\alpha\rangle$ and $|\beta\rangle$. If there are the same, we're guaranteed to get 0 (i.e. 'near'), and if they are orthogonal, we get 1 with probability $1/2$ (likely to say 'far apart').

This has applications to the graph isomorphism problem as follows. Letting $f_A(\pi) = \pi(A)$, we can, as usual, build the state

$$|\xi_A\rangle = \sum_{\pi \in \mathcal{P}_n} |\pi\rangle|\pi(A)\rangle .$$

Suppose we can 'forget' the contents of the first register (this is impossible in practice, due to the superposition). Then we'd be left with:

$$|\eta_A\rangle = \sum_{\pi \in \mathcal{P}_n} |\pi(A)\rangle .$$

Build the same state $|\eta_B\rangle$ for another graph B which we want to compare to A for isomorphism.

If A is isomorphic to B, then it's clear that $|\eta_A\rangle = |\eta_B\rangle$. If A is not isomorphic to B, we'd have

$$\langle\eta_B|\eta_A\rangle = \sum_{\substack{\pi \in \mathcal{P}_n \\ \sigma \in \mathcal{P}_n}} \langle\pi(B)|\sigma(A)\rangle = 0,$$

since $\pi(B) \neq \sigma(A)$ for any π, σ, else A and B would be isomorphic via $\pi^{-1}\sigma(A) = B$.

Thus we can apply the swap test to $|\eta_B\rangle$ and $|\eta_A\rangle$ to tell if the graphs are isomorphic or non-isomorphic. Repeat a constant number of times to reach any level of accuracy.

Example (Partial balanced vs constant): Let B_n be the set of all n-bit strings and let $S \subseteq B_n$ with $|S|$ even. Suppose we have an oracle $f : B_n \to B_1$ such that f restricted to S is either balanced (half values 0, half 1) or constant (all 0 or all 1). We are given the uniform superposition state on S:

$$|\alpha\rangle = \frac{1}{\sqrt{|S|}} \sum_{x \in S} |x\rangle ,$$

and a quantum oracle U_f for f. The *partial balanced vs constant problem* is to determine whether f restricted to S is balanced or constant.

In the case $S = B_n$, this is solved by the *Deutsch-Jozsa algorithm*. We add the ancilla $|1\rangle$ to $|\alpha\rangle$ to get $|1\rangle|\alpha\rangle$. We apply the Hadamard gate H to $|1\rangle$ and pass the result to U_f. We observed the answer is of the form $|\xi\rangle|-\rangle$, and discard the second register at this point. We notice that the possible $|\xi\rangle$ for constant or balanced are orthogonal, so rotating back to the standard basis using H^n, they remain orthogonal. We measure: if we see $000...0$, the function was constant, and if we see anything else it was balanced.

The case when $S \subset B_n$ can help us solve graph isomorphism. Define $\sigma \in \mathcal{P}_{2n}$ to be the permutation swapping L_A and L_B in their listed order, and consider the group $G = \mathcal{P}_n \times \mathcal{P}_n \cup \sigma(\mathcal{P}_n \times \mathcal{P}_n)$. Note that for the graph $C = A \cup B$ above, we have $\text{Aut}(C) \leq G$.

Suppose we are given the function $f_C(\pi) = \pi(C)$, as usual, and a quantum oracle for f_C. Begin by making

$$\frac{1}{\sqrt{|\mathcal{P}_{2n}|}} \sum_{\pi \in G} |\pi\rangle|0\rangle .$$

Applying U_{f_C}, measuring, and discarding the second register, we're left with a random coset state:

$$|\rho\text{Aut}(C)\rangle = \frac{1}{\sqrt{|\text{Aut}(C)|}} \sum_{\pi \in \text{Aut}(C)} |\rho\pi\rangle .$$

We can write out $\rho\pi$ in terms of bit strings to get a partial balanced versus constant problem setup. Then an algorithm for solving partial balanced versus constant would solve graph isomorphism, since we'd be able to use the function $f : G \to B_1$, defined by $f(\pi) = 0$ if $\pi \in \mathcal{P}_n \times \mathcal{P}_n$ and $f(\pi) = 1$ if $\pi \in \sigma(\mathcal{P}_n \times \mathcal{P}_n)$.

From our above work, we know that $\text{Aut}(C)$ is 50/50 in $\mathcal{P}_n \times \mathcal{P}_n$ and $\sigma(\mathcal{P}_n \times \mathcal{P}_n)$ when A is isomorphic to B (i.e. f is *balanced*), and $\text{Aut}(C)$ is entirely in $\mathcal{P}_n \times \mathcal{P}_n$ when A is no isomorphic to B (i.e. f is constant).

3 Phase estimation

3.1 Problem statement and setup

Definition: Suppose we are given a unitary operator U and an eigenstate $|v_\phi\rangle$ with $U|v_\phi\rangle = e^{2\pi i\phi}|v_\phi\rangle$. WLOG we may assume $0 \le \phi < 1$. The *phase estimation problem* is to determine the first n (for any n) binary digits of ϕ:

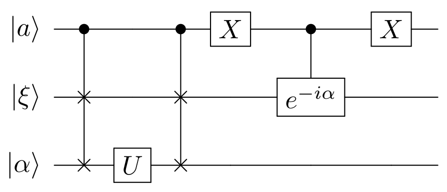
$$\phi \approx 0.i_1 i_2 ... i_n = \frac{i_1}{2} + \frac{i_2}{4} + ... + \frac{i_n}{2^n}.$$

To setup, we first need *controlled U^k gates*:

Definition: The *controlled U^k gate* is the unitary gate defined by $c\text{-}U^k|0\rangle|\xi\rangle = |0\rangle|\xi\rangle$ and $c\text{-}U^k|1\rangle|\xi\rangle = |1\rangle U^k|\xi\rangle$. Note that $c\text{-}(U^k)$ (the controlled U^k gate) is the same as $(c\text{-}U)^k$ (controlled U, but k times).

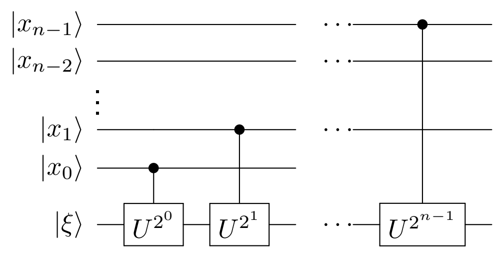
There are immediately questions about implementation of such gates. If U is given as a formula or circuit description, it's easy to get $c - U$ since we can just control each gate in the circuit separately.

If U is given as a blackbox, we need more information. It suffices to have an eigenstate $|\alpha\rangle$ of U with known eigenvalue $e^{i\alpha}$: $U|\alpha\rangle = e^{i\alpha}|\alpha\rangle$. Then the circuit:



effects $|a\rangle|\xi\rangle|\alpha\rangle \to (c\text{-}U)|a\rangle|\xi\rangle$. This is easy to check by eye, just running through $|a\rangle = |0\rangle, |1\rangle$ separately. The $e^{-i\alpha}$ box just multiplies the whole state by $e^{-i\alpha}$. The two crosses are controlled SWAPs between the lower qubits.

For our purposes, we'll want a '*generalised controlled U*', with $(c\text{-}U^x)|x\rangle|\xi\rangle = |x\rangle U^x|\xi\rangle$ for $x \in \mathbb{Z}_{2^n}$. Note this generalised controlled U is actually different because it doesn't just act on a single qubit, but on n qubits. This can be implemented using the single-qubit controlled U gates via the circuit:



Here, we have written $|x\rangle = |x_{n-1}\rangle|x_{n-2}\rangle ... |x_0\rangle$ in its full binary expansion, where

$$x = 2^{n-1}x_{n-1} + 2^{n-2}x_{n-2} + ... + 2x_1 + x_0.$$

Again, it's easy to just check that this circuit does what it's supposed to. Note in particular that if $|\xi\rangle = |v_\phi\rangle$, then the output is $e^{2\pi i\phi x}|x\rangle|v_\phi\rangle$.

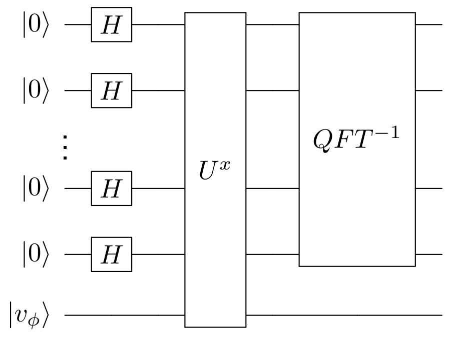
3.2 The phase estimation algorithm

Algorithm (Phase estimation): We are given $|v_\phi\rangle$ and the relevant controlled U gates.

1. Start with $|00...0\rangle|v_\phi\rangle$. Make the uniform superposition from the first n qubits: $\frac{1}{\sqrt{2^n}}\sum_{x\in B_n}|x\rangle|v_\phi\rangle$.
2. Apply U^x to get $\frac{1}{\sqrt{2^n}}\sum_{x\in B_n}e^{2\pi i\phi x}|x\rangle|v_\phi\rangle$.
3. Apply the inverse quantum Fourier transform to the first n qubits. Measure the first n qubits to see some $y_0 y_1 ... y_{n-1}$. Then output the answer:

$$0.y_1 y_2 ... y_{n-1} = \frac{y_0}{2} + \frac{y_1}{4} + ... + \frac{y_{n-1}}{2^{n-1}}.$$

As a circuit diagram, the procedure is:



followed by a measurement of the first n qubits. The top n qubits are often referred to as *lines* in phase estimation.

Our immediate question should be: does it work? This is hard to answer when ϕ isn't exact to n binary places. However, in the case that $\phi = 0.z_0 z_1 ... z_{n-1} = z/2^n$, where $z = z_0 z_1 ... z_{n-1}$, before we take the Fourier transform, we have the state in the first n registers:

$$\frac{1}{\sqrt{2^n}}\sum_x e^{2\pi i x z/2^n}|x\rangle = QFT|z\rangle.$$

Hence applying the inverse QFT gives $|z\rangle$, and hence measurement gives z exactly! So it works in the exact case.

In other cases, the answer $0.y_0y_1...y_{n-1}$ is just an approximation; we'll have to decide how good it is.

REMARK: In many algorithms, it is useful to use everything in the phase estimation algorithm *except* for the final measurement. This gives a *unitary operation* $|000...0\rangle |v_\phi\rangle \mapsto |z_0\rangle ... |z_{n-1}\rangle |v_\phi\rangle$ in the exact case, where the first n registers contain the information about the eigenvalue. This operation is sometimes written U_{PE}.

3.3 Can we even do this?

Quantum gates are defined *up to a phase*, thus it seems like it's impossible to determine ϕ. We can just redefine the gate U with $\tilde{U} = e^{i\alpha}U$, and we wouldn't be able to notice anything different physically. So phase estimation seems fruitless.

However, phase estimation doesn't actually use the gates U and $\tilde{U}$, but their controlled analogues. The gate $e^{i\alpha}c\text{-}U$ has a gap in the output phase between U being on and off, given by $e^{2\pi i\phi}$. Hence we can detect this *phase difference*.

3.4 The phase estimation theorem

It's now time to analyse how good *approximate* phase estimation is. This is enshrined in the Theorem:

Theorem (Phase estimation): If the measurements in the phase estimation algorithm give an answer $\theta = 0.y_0y_1...y_{n-1}$, then

(i) Prob(θ is the closest n-binary digit approximation to ϕ)

$$\geq \frac{4}{\pi^2};$$

This is the probability that *all* of our n lines are 'good', i.e. they are all significant.

(ii) Prob$(|\theta-\phi| \geq \epsilon) \leq \dfrac{1}{2^{n+1}\epsilon}$. This means that if we want ϕ accurate to m digits (i.e. $\epsilon = 1/2^m$) with probability $1-\eta < 1$, then:

$$\text{Prob}\left(|\theta-\phi| \geq \frac{1}{2^m}\right) \geq \frac{2^m}{2^{n+1}},$$

so it's sufficient to ask $2^m/2^{n+1} < \eta$, i.e.

$$n > m + \log\left(\frac{1}{\eta}\right) + 1 = O(m).$$

So to get exponentially higher accuracy (i.e. more digits), need only polynomially more lines! That is, each line added to the algorithm makes it more likely to be accurate to another digit.

Proof: Final state in the algorithm before measurement is

$$\frac{1}{2^n}\sum_{y\in B_n}\left(\sum_{x\in B_n} e^{2\pi i(\phi-y/2^n)x}\right)|y\rangle\,.$$

Write $\delta(y) = \phi - y/2^n$ (the amount we differ from the true answer with our guess). Then the probability of seeing $y = y_0y_1...y_{n-1}$ on measurement is

$$\text{Prob}(y = y_0y_1...y_{n-1}) = \frac{1}{2^{2n}}\left|\sum_{x=0}^{2^n-1} e^{2\pi i\delta(y)x}\right|^2.$$

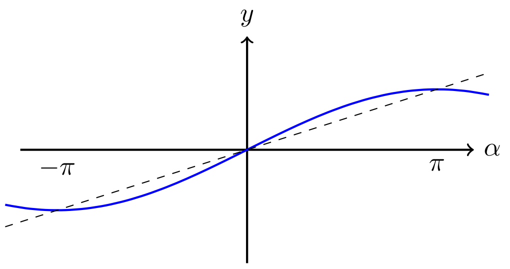
Summing the geometric progression inside the modulus, we have

$$\text{Prob}(y = y_0y_1...y_{n-1}) = \frac{1}{2^{2n}}\left|\frac{1-e^{2^n\cdot 2\pi i\delta(y)}}{1-e^{2\pi i\delta(y)}}\right|^2.$$

(i) Let $y = a = a_0a_1...a_{n-1}$ be the closest n-bit approximation to $2^n\phi$. Then $|\phi - a/2^n| \leq 1/2^{n+1}$ (easiest to see by drawing a diagram comparing $a/2^n$ and $(a-1)/2^n$ relative to ϕ). Hence $|\delta(a)| \leq 1/2^{n+1}$.

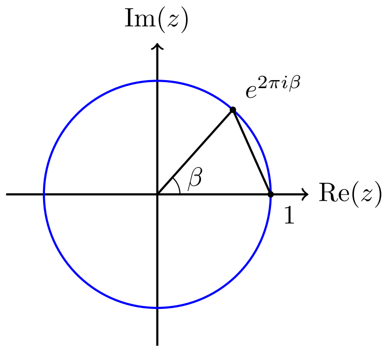
We bound the probability using the following:

(a) $|1-e^{i\alpha}| = |2i\sin(\frac{\alpha}{2})| \geq \frac{2}{\pi}|\alpha|$ if $|\alpha| \leq \pi$. The proof of this bound is to look at the picture:



where the dashed line is $y = \alpha/\pi$ and the blue line is $y = \sin(\alpha/2)$.

(b) $|1-e^{2\pi i\beta}| \leq 2\pi\beta$. The proof of this bound is to look at the picture:



The length of the arc is $2\pi\beta$. The length of the chord is $|e^{2\pi i\beta} - 1|$, and the inequality follows.

Applying these inequalities to lower bound our probability, using (a) on the numerator and (b) on the denominator, we see that

$$\text{Prob}(y = y_0 y_1 ... y_{n-1}) \geq \frac{4}{\pi^2},$$

as required.

(ii) Now try to *upper bound* our probability. Use $|1-e^{i\alpha}| \leq 2$ on the denominator, and use (a) again on the numerator to deduce

$$\text{Prob}(y = y_0 y_1 ... y_{n-1}) \leq \frac{1}{2^{n+2}\delta(y)^2}.$$

To get the inequality in (ii), we must sum this for all $|\delta(y)| > \epsilon$. We know that the $\delta(y)$ values are spaced by $1/2^n$; let δ_+ be the first $\delta(y)$ with $\delta(y) \geq \epsilon$, and let δ_- be the first $\delta(y)$ with $\delta(y) \leq -\epsilon$.

Certainly $|\delta_+|, |\delta_-| \geq \epsilon$. If $|\delta(y)| \geq \epsilon$, we can then write

$$\delta(y) = \delta_+ + \frac{k}{2^n}, \text{ or } \delta(y) = \delta_- - \frac{k}{2^n}.$$

Hence: $|\delta(y)| \geq \epsilon + \dfrac{k}{2^n}$ in both cases. Hence:

$$\text{Prob}(|\delta(y)| > \epsilon) \leq 2\sum_{k=0}^{\infty} \frac{1}{2^{2n+2}} \left(\frac{1}{\epsilon + \frac{1}{2^n}} \right)^2$$

$$\leq \frac{1}{2}\int_0^\infty \frac{dk}{(2^n\epsilon + k)^2} = \int_{2^n\epsilon}^\infty \frac{dk}{k^2} = \frac{1}{2^{n+1}\epsilon}.$$

Note by going to ∞, we weakened the bound, but made it easier to calculate! □

3.5 Time complexity of phase estimation

If $c\text{-}U^{2^k}$ is implemented as $(c\text{-}U)^{2^k}$, then the phase estimation algorithm needs exponential time, since its uses

$$1 + 2 + ... + 2^{n-1} = 2^n - 1$$

$c\text{-}U$ gates. *However*, for some special U's, $c\text{-}U^{2^k}$ can be implemented in polynomial time, $O(\text{poly}(k))$. This mirrors the calculation of x^{2^n} via repeated squaring, rather than $x \cdot x \cdot ... \cdot x$, but only works for some unitary U.

3.6 Applications of phase estimation

Example 1 (Implementing Mth roots): Let $U^{1/M}$ be the principal Mth root of the unitary gate U, defined to have the same eigenstates as U and corresponding eigenvalues $e^{2\pi i\phi/M}$, where U has eigenvalues $e^{2\pi i\phi}$. Suppose $\phi = y/2^n$ for some $0 \leq y \leq 2^n$, $y \in \mathbb{Z}$.

Suppose also that we can implement the *phase gate* $P(\alpha) = \text{diag}\{(1, e^{i\alpha}\}$ for any α, and any controlled $c\text{-}U$ gate we wish, and its inverse $c\text{-}U^{-1}$. Then it's possible to implement $U^{1/M}|\xi\rangle$.

To do so, let

$$|\xi\rangle = \sum_\phi \beta_\phi |v_\phi\rangle,$$

where $|v_\phi\rangle$ are the eigenvectors of U, with $U|v_\phi\rangle = e^{2\pi i\phi}|v_\phi\rangle$. Adjoint $|00...0\rangle$ to the state, and apply the unitary phase estimation operation to this state to get:

$$U_{\text{PE}}|\xi\rangle = \sum_\phi \beta_\phi |2^n\phi\rangle |v_\phi\rangle.$$

Apply the sequence of phase gates now. Apply $P(2\pi/2M)$ to the first qubit, $P(2\pi/4M)$ to the second, ... , $P(2\pi/2^n M)$ to the last. If $2^n\phi = i_1 ... i_n$ in binary, we have

$$\frac{2\pi\phi}{M} = i_1\frac{2\pi}{2M} + i_2\frac{2\pi}{4M} + \cdots + i_n\frac{2\pi}{2^n M}.$$

Hence the overall phase created is $e^{2\pi i\phi/M}$, next to $|2^n\phi\rangle$. Thus we're left with the state

$$\sum_\phi \beta_\phi e^{2\pi i\phi/M} |2^n\phi\rangle |v_\phi\rangle.$$

Apply U_{PE}^{-1} to get rid of the $|2^n\phi\rangle$ state, and turn it into a $|00...0\rangle$. It's then safe to discard it, and we're left with

$$\sum_\phi \beta_\phi e^{2\pi i\phi/M} |v_\phi\rangle,$$

which is the desired outcome.

Example 2 (Implementing non-unitary gates): Let A be an n-qubit Hermitian operator with distinct eigenvalues $\lambda_i = c_i/2^n$, for c_i an integer in $0 \leq c_i < 2^n$. Suppose that for the unitary operations $U_\pm = e^{\pm 2\pi i A}$ we can implement $c\text{-}U_\pm$. Let $|b\rangle$ be given to us (but unknown) and suppose we want the normalised state $A|b\rangle$ with some non-zero probability.

Suppose also that we can perform controlled rotations of the form

$$|c\rangle|0\rangle \to |c\rangle(\cos(\theta_c)|0\rangle + \sin(\theta_c)|1\rangle),$$

where $\sin(\theta_c) = c/2^n$ and $0 \leq c < 2^n$ is an integer. Then we can achieve our goal as follows.

Write $|b\rangle = \sum_j \beta_j |u_j\rangle$, where $|u_j\rangle$ are the eigenvectors of A with eigenvalues λ_i. Then applying U_{PE} to $|00...0\rangle|b\rangle$, we obtain the state:

$$\sum_j \beta_j |c_j\rangle |u_j\rangle.$$

Squeeze an ancilla qubit $|0\rangle$ into the middle of these, and apply the controlled rotation to the first two registers to get:

$$\sum_j \beta_j |c_j\rangle |0\rangle |u_j\rangle \to$$
$$\sum_j \beta_j \cos(\theta_{c_j}) |c_j\rangle |0\rangle |u_j\rangle + \beta_j \sin(\theta_{c_j}) |c_j\rangle |1\rangle |u_j\rangle$$
$$= \sum_j \beta_j \sqrt{1-\lambda_j^2} |c_j\rangle |0\rangle |u_j\rangle + \beta_j \lambda_j |c_j\rangle |1\rangle |u_j\rangle$$

Now *measure* the middle register. This is called *post-selection* and we'll see examples of it later in the course. If we see 1, everything's fine, just apply U_{PE}^{-1} and we're done. If we see a 0, just start all over again.

Let λ_1 be the eigenvalue with the smallest square. The probability we'll see a 1 is then

$$||\sum_j \beta_j \lambda_j |c_j\rangle |u_j\rangle||^2 = \sum_j |\beta_j|^2 |\lambda_j|^2 \geq |\lambda_1|^2 \sum_j |\beta_j|^2 = \lambda_1^2.$$

So we'll succeed with a probability exceeding that of the square of the smallest eigenvalue.

Example 3 (Implementing the QFT): Suppose we want to implement QFT_Q for $2^{m-1} < Q < 2^m$. We use the following method, called *Kitaev's method*. Let

$$|a\rangle = QFT_Q |a\rangle = \frac{1}{\sqrt{Q}} \sum_{b=0}^{Q-1} \omega^{ab} |b\rangle ,$$

where $\omega = e^{2\pi i/Q}$. Then it suffices to implement $|a\rangle \to |\eta_a\rangle$, and we're done by linearity.

We'll achieved this in two chunks, by considering a mapping on $\mathcal{H}_Q \otimes \mathcal{H}_Q$:

$$|a\rangle |0\rangle \underbrace{\to}_{\text{Step 1}} |a\rangle |\eta_a\rangle \underbrace{\to}_{\text{Step 2}} |0\rangle |\eta_a\rangle .$$

STEP 1: Start with $|00...0\rangle$, and make the uniform superposition

$$|\psi_0\rangle = \frac{1}{\sqrt{M}} \sum_{x=0}^{2^m-1} |x\rangle .$$

Consider the classically-computable function

$$f(x) = \begin{cases} 0 & x < Q \\ 1 & x \geq Q. \end{cases}$$

Since it is classically efficient to compute this, we can efficiently implement U_f. Pass $|\psi_0\rangle |0\rangle$ to U_f to get

$$\frac{1}{\sqrt{M}} \sum_{x=0}^{2^n-1} |x\rangle |f(x)\rangle .$$

Measure the second register. The chance of seeing 0 is greater than $1/2$, since Q is more than halfway along $2^{m-1} < Q < 2^m$. If we fail, just try again. We get K fails with exponentially small probability $1/2^K$.

When we get 0, the state reduces to

$$|\xi\rangle = \frac{1}{\sqrt{Q}} \sum_{b=0}^{Q-1} |b\rangle .$$

We now add $|a\rangle$ in to get $|a\rangle |\xi\rangle$. To get to $|a\rangle |\eta_a\rangle$ from here, we want a unitary operation V that effects $V |a\rangle |b\rangle = \omega^{ab} |a\rangle |b\rangle$, which gives $V |a\rangle |\xi\rangle = |a\rangle |\eta_a\rangle$.

To implement V, we again do things in chunks. Consider the operation $U |b\rangle = \omega^b |b\rangle$. Noticing that $\omega^b = \omega^{b_{m-1}2^{m-1}} ... \omega^{b_0 2^0}$, when b is written in binary, we see that

$$U = P(\omega^{2^{m+1}}) \otimes ... \otimes P(\omega^{2^0}),$$

where $P(\alpha)$ are the phase gates from before. So U is efficiently implementable. We can obtain $c\text{-}U$ because we have an explicit circuit for U, and as in phase estimation, we can build $c\text{-}U^a$, a generalised controlled U gate. This effects:

$$c\text{-}U^a(|a\rangle |b\rangle) = |a\rangle U^a |b\rangle = \omega^{ab} |a\rangle |b\rangle .$$

So V is implementable, and we're done. We can do Step 1.

STEP 2: Let U be some unitary operation with eigenvalues $\omega^a = e^{2\pi i a/Q}$ and eigenstates $|\eta_a\rangle$. Then phase estimation of U with U_{PE} gives

$$U_{\text{PE}} |0\rangle |\eta_a\rangle = |a\rangle |\eta_a\rangle .$$

(We're playing fast and loose here, and assuming phase estimation is exact. The non-exactness can be accounted for in practice.) Then U_{PE}^{-1} would achieve Step 2!

We just need U. Use U defined by $U |x\rangle = |x - 1 \text{ (mod } Q)\rangle$; we can check that $U |\eta_a\rangle = \omega^a |\eta_a\rangle$ because these are really shift invariant states. Note also U is clearly implementably in polynomial time.

The non-exactness in phase estimation above actually results in an operation:

$$U_{\text{PE}} |0\rangle |\eta_a\rangle = \left(\sqrt{1-\epsilon} |a\rangle + \sqrt{\epsilon} |a^\perp\rangle\right) |\eta_a\rangle ,$$

where we are using $O(\log(1/\epsilon))$ lines in phase estimation, for any small ϵ. Then the difference between the actual state and the non-exact one is:

$$||U_{\text{PE}}^{-1} |a\rangle |\eta_a\rangle - |0\rangle |\eta_a\rangle || = O(\sqrt{\epsilon}).$$

Thus we can still approximate QFT_Q to any desired precision.

4 Amplitude amplification

4.1 Reflection operators

Notation: Write $L_\alpha = \text{span}\{|\alpha\rangle\}$, and $L_\alpha^\perp$ for the orthogonal complement of span$\{|\alpha\rangle\}$.

Definition: The *reflection operator* in $L_\alpha^\perp$ is defined by $I_{|\alpha\rangle} = I - 2|\alpha\rangle\langle\alpha|$.

Theorem: We have

(i) $I_{|\alpha\rangle}|\alpha\rangle = -|\alpha\rangle$ and $I_{|\alpha\rangle}|\beta\rangle = |\beta\rangle$ for $|\beta\rangle \in L_\alpha^\perp$;

(ii) for unitary U, $UI_{|\alpha\rangle}U^\dagger = I_{U|\alpha\rangle}$.

Proof: (i) Obvious from $\langle\alpha|\beta\rangle = 0$. (ii) Trivial. □

Definition: Let A be spanned by the orthonormal basis $\{|a_1\rangle, ..., |a_k\rangle\}$. Define the *projection operator onto* A by

$$P_A = \sum_{i=1}^{k} |a_i\rangle\langle a_i|.$$

Definition: Define the *reflection operator* in $A^\perp$ (the orthogonal complement of A) to be $I_A = I - 2P_A$.

Theorem: If $|a\rangle \in A$, then $I_A|a\rangle = -|a\rangle$. If $|b\rangle \in A^\perp$, then $I_A|b\rangle = |b\rangle$.

Proof: Obvious from $\langle a|b\rangle = 0$. □

4.2 Review of Grover's algorithm

Definition: Let B_n be the set of n-bit strings. We are given a function $f : B_n \to B_1$ such that $f(x_0) = 1$ for a unique 'good' $x_0 \in B_n$, and $f(x) = 0$ otherwise. We are also given the quantum implementation of f, U_f. *Grover's problem* is to determine x_0 with high probability.

The solution of Grover's problem is given by *Grover's algorithm*. The setup is as follows.

STEP 1: First, we need to implement $I_{|x_0\rangle}$. We can get $I_{|x_0\rangle}|x\rangle$ for any basis state $|x\rangle$ by querying the oracle once with $|x\rangle|-\rangle$. This gives:

$$U_f|x\rangle|-\rangle = |x\rangle\left(\frac{|f(x)\rangle - |1\oplus f(x)\rangle}{\sqrt{2}}\right).$$

So if $|x\rangle = |x_0\rangle$, we have $f(x) = 1$, and the RHS is $-|x\rangle|-\rangle$. Discard the second qubit. If $|x\rangle \neq |x_0\rangle$, we have $f(x) = 0$, and the RHS is $|x\rangle|-\rangle$. Again, discard the second qubit.

This can be extended to all states by linearity.

STEP 2: We must define the *Grover iteration operator* and determine how it acts.

Definition: The *Grover iteration operator* is defined by

$$Q = -H^n I_{|0\rangle} H^n I_{|x_0\rangle} = -I_{|\psi_0\rangle} I_{|x_0\rangle},$$

where $H^n = H \otimes ... \otimes H$ and $|\psi_0\rangle = H^n|0\rangle$ is the uniform superposition.

Each step of the Grover iteration operator is implementable (indeed, $I_{|0\rangle}$ can be implemented using the classically efficient function $f : B_n \to B_1$ given by $f(0) = 1$ and $f(x) = 0$ otherwise).

Notice also that each application of Q uses one query to U_f. So we need to count the number of times we used Q at the end of any algorithm.

Grover's Theorem: In the 2-dimensional span of $|\psi_0\rangle$ and $|x_0\rangle$, the action of Q is a rotation by angle 2α, where

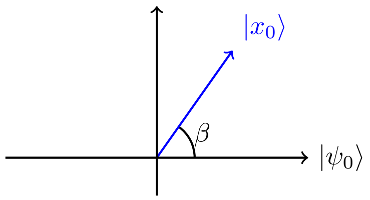
$$\sin(\alpha) = \frac{1}{\sqrt{N}}.$$

Proof: Easiest to act on an orthonormal basis in this span and determine the matrix of the operator. It will be of rotation form. □

Grover's Algorithm: To find x_0 given U_f:

1. Make $|\psi_0\rangle$, the uniform superposition.
2. Apply Q m time to $|\psi_0\rangle$, where
$$m = \frac{\arccos(1/\sqrt{N})}{2\arcsin(1/\sqrt{N})}.$$
3. Measure to see x_0 with high probability.

The proof this works is by picture:



Notice that the angle in the figure obeys $\cos(\beta) = \langle x_0|\psi_0\rangle = 1/\sqrt{N}$ (in particular, both $|x_0\rangle$ and $|\psi_0\rangle$ are normalised). Since we start at $|\psi_0\rangle$, and Q acts as an anticlockwise rotation by 2α, we need

$$m = \frac{\beta}{2\alpha} = \frac{\arccos(1/\sqrt{N})}{2\arcsin(1/\sqrt{N})}$$

iterations of Q to get as close to $|x_0\rangle$ as possible. The measurement gives x_0 with high probability, since if we are at state $|\psi\rangle$, within $\pm\alpha$ of $|x_0\rangle$, the amplitude for seeing x_0 is $\langle\psi|x_0\rangle \approx \cos(\alpha)$. So probability is $\cos^2(\alpha) = 1 - \sin^2(\alpha) = 1 - \frac{1}{N}$, which is nearly 1 for large N.

We can also work out the time complexity of the algorithm by considering the large N limit.

Theorem: Grover's algorithm has complexity $O(\sqrt{N})$.

Proof: For large N, $\arccos(1/\sqrt{N}) \approx \pi/2$ (using $\cos(\theta) \approx 1$ for small θ), and $\arcsin(1/\sqrt{N}) \approx 1/\sqrt{N}$ (using $\sin(\theta) \approx \theta$ for small θ). Therefore in the large N limit,

$$m = \frac{\pi\sqrt{N}}{4}.$$

So we use U_f $O(\sqrt{N})$ times. □

In particular, the naïve classical approach of checking everything takes $O(N)$ time, so we get a square-root speed-up quantumly. In fact, it's possible to show:

Theorem: Grover's algorithm is optimal for unique element search.

Proof: Not required. □

4.3 Amplitude amplification

Notation: Let G be a 'good' subspace of a state space $\mathcal{H}$ and let $G^\perp$ be its orthogonal complement, which we call the 'bad' subspace, so that $\mathcal{H} = G \oplus G^\perp$.

Theorem: For any state $|\psi\rangle \in \mathcal{H}$, we can write $|\psi\rangle = \sin(\theta)|g\rangle + \cos(\theta)|b\rangle$, where $|g\rangle \in G$, $|b\rangle \in G^\perp$.

Proof: We must be able to write $|\psi\rangle = c|g\rangle + d|b\rangle$ for complex c and d satisfying $|c|^2 + |d|^2 = 1$. Let $c = e^{i\theta}c'$ and $b = e^{i\phi}b'$ for $c', b' \in G$. Then $|\psi\rangle = e^{i\theta}c'|g\rangle + e^{i\phi}b'|b\rangle = e^{i\theta}(c'|g\rangle + e^{i\phi - i\theta}b'|b\rangle)$.

Note that the overall phase can be removed giving the same quantum state. Note that $e^{i\phi-i\theta}|b\rangle \in G^\perp$, so can just redefine $|b\rangle$ to get real coefficients, giving result. □

In our algorithm, we'll use the reflection operator $I_{|\psi\rangle} = I - 2|\psi\rangle\langle\psi|$ and the projection operator $I_G = I - 2P_G$, as defined above. In particular, notice that

$$\sin(\theta) = ||P_G|\psi\rangle||.$$

Definition: Define the *generalised Grover operator* by $Q = -I_{|\psi\rangle}I_G$.

Amplitude Amplification Theorem: In the 2-dimensional subspace spanned by $|g\rangle$ and $|\psi\rangle$ (or equivalently by the orthonormal vectors $|g\rangle$ and $|b\rangle$), Q is a rotation by 2θ, where $\sin(\theta) = ||P_G|\psi\rangle|| =$ length of good projection of $|\psi\rangle$.

Proof: Note that $I_G|g\rangle = -|g\rangle$ and $I_G|b\rangle = -|b\rangle$. Therefore

$$Q|g\rangle = I_{|\psi\rangle}|g\rangle, \quad Q|b\rangle = -I_{|\psi\rangle}|b\rangle.$$

Now compute action of $I_{|\psi\rangle} = I - 2|\psi\rangle\langle\psi|$. Writing its definition out in full, we have

$$\begin{aligned} I_{|\psi\rangle} &= I - 2\sin^2(\theta)|g\rangle\langle g| - 2\sin(\theta)\cos(\theta)|g\rangle\langle b| \\ &\quad -2\sin(\theta)\cos(\theta)|b\rangle\langle g| - 2\cos^2(\theta)|b\rangle\langle b|. \end{aligned}$$

Therefore $Q|g\rangle = (1 - 2\sin^2(\theta))|g\rangle - 2\sin(\theta)\cos(\theta)|b\rangle = \cos(2\theta)|g\rangle - \sin(2\theta)|b\rangle$ and $Q|b\rangle = 2\sin(\theta)\cos(\theta)|g\rangle - 2(1 - 2\cos^2(\theta))|b\rangle = \sin(2\theta)|g\rangle + \cos(2\theta)|b\rangle$.

Hence Q has matrix $\begin{pmatrix} \cos(2\theta) & -\sin(2\theta) \\ \sin(2\theta) & \cos(2\theta) \end{pmatrix}$ in the basis $\{|g\rangle, |b\rangle\}$. □

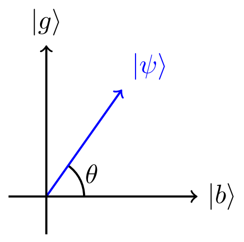
From the above Theorem, it follows immediately that n applications of Q to $|\psi\rangle$ give:

$$Q^n|\psi\rangle = \sin((2n+1)\theta)|g\rangle + \cos((2n+1)\theta)|b\rangle.$$

If we measure $Q^n|\psi\rangle$ for good versus bad, then the probability we see a good value is Prob(see good) $= \sin^2((2n+1)\theta)$. This is maximised when $(2n+1)\theta = \pi/2$, i.e.

$$n = \frac{\pi}{4\theta} - \frac{1}{2} = \frac{(\pi/2 - \theta)}{2\theta}.$$

This result is also clear from the diagram:



Example: If $\theta = \pi/6$ to start with, then $n = 1$, exactly and the probability is 1. So one iteration of Q is sufficient to guarantee seeing a good value with certainty on measurement.

In general, n is not an integer, so we must use the nearest integer. For θ small, this is: $n =$

$$\left[\frac{\pi}{2\theta} - \frac{1}{2}\right] \approx \left[\frac{\pi}{4\theta}\right] = O\left(\frac{1}{\theta}\right) = O\left(\frac{1}{||\text{good projection of } |\psi\rangle||}\right).$$

Furthermore, $Q^n|\psi\rangle$ will be within angle $\pm\theta$ of $|g\rangle$, so it follows that Prob(see good) $\geq \cos^2(\theta) \approx 1 - O(\theta^2)$, as for Grover's algorithm. So if θ is small, seeing good has a very high chance.

Remarks: 1. Note the relative amplitudes of good labels in $|g\rangle$ stay the same as they were in $|\psi\rangle$ throughout.

2. The measurement is generally probabilistic, but *if* $\sin(\theta)$ is *known*, then it can be made exact. See later.

4.4 Implementation of ampl. ampl.

To implement ampl. ampl., we need to be able to implement $-Q = I_{|\psi\rangle} I_G$.

Theorem: I_G can be implemented in poly(n) time.

Proof: The classical indicator function

$$f(x) = \begin{cases} x & \text{is good,} \\ x & \text{is bad,} \end{cases}$$

Is clearly implementable in polynomial time. Therefore the associated quantum gate, U_f, can be implemented in polynomial time. To implement I_G on $|x\rangle$, adjoint an ancilla $|-\rangle$ and note that

$$U_f |x\rangle |-\rangle = \frac{|x\rangle |0 \oplus f(x)\rangle - |x\rangle |1 \oplus f(x)\rangle}{\sqrt{2}} = (-1)^{f(x)} |x\rangle |-\rangle .$$

Discard the $|-\rangle$, and we've implemented I_G. □

In most quantum algorithms, we start with $|\psi\rangle = |\psi_0\rangle$, the uniform superposition, e.g. as in Grover's algorithm. So we'll just show that $I_{|\psi\rangle\rangle}$ can be implemented efficiently.

Theorem: $I_{|\psi_0\rangle}$, where $|\psi_0\rangle$ is the uniform superposition, can be implemented in $O(n)$ time.

Proof: Define the 3-qubit *Toffoli gate* by

$$T_{123} |a\rangle_1 |b\rangle_2 |c\rangle_3 = \begin{cases} |a\rangle_1 |b\rangle_2 X |c\rangle_3 & \text{if } a = b = 1, \\ |a\rangle_1 |b\rangle_2 |c\rangle_3 & \text{otherwise.} \end{cases}$$

It's easy to see that T_{123} is the quantum oracle U_g for the function $g(a,b) = ab$ (just observe they agree on a basis).

We now observe that $2n-2$ Toffoli gates and one CX gate can be used to implement the transformation

$$|c_1\rangle_1 \ldots |c_n\rangle_n \underbrace{|0\rangle_{w_1} \ldots . |0\rangle_{w_{n-1}}}_{\text{'work space'}} |y\rangle_t \to$$
$$|c_1\rangle_1 \ldots |c_n\rangle_n |0\rangle_{w_1} \ldots |0\rangle_{w_{n-1}} |y \oplus c_1 c_2 \ldots c_n\rangle_t .$$

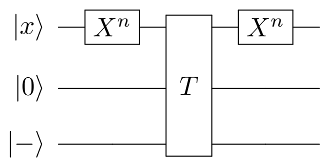
The procedure we use is as follows:

1. Apply T_{12w_1} to get
$$|c_1\rangle_1 |c_2\rangle_2 \ldots |c_n\rangle_n |c_1 c_2\rangle_{w_1} |0\rangle_{w_2} \ldots |0\rangle_{w_{n-1}} |y\rangle_t .$$
2. Next apply $T_{3w_1w_2}$ to get
$$|c_1\rangle_1 \ldots |c_n\rangle_n |c_1 c_2\rangle_{w_1} |c_1 c_2 c_3\rangle_{w_2} |0\rangle_{w_3} \ldots |0\rangle_{w_{n-1}} |y\rangle_t .$$
3. Continue in this fashion, applying $T_{4w_2w_3}$, then $T_{5w_3w_4}$ up to $T_{nw_{n-2}w_{n-1}}$. Then the final state is:
$$|c_1\rangle_1 \ldots |c_n\rangle_n |c_1 c_2\rangle_{w_1} \ldots |c_1 c_2 \ldots c_n\rangle_{w_{n-1}} |y\rangle_t .$$
4. Apply CX to the final two qubits to get $|y \oplus c_1 \ldots c_n\rangle_t$ as the target qubit. We can then erase the work space by applying $T_{nw_{n-2}w_{n-1}}$ again, down to T_{12w_1}, i.e. apply all the Toffoli gates in reverse order.

Indeed, this procedure takes $2n-2$ Toffolis and one CX. We now build a circuit for $I_{|\psi_0\rangle}$. First, note that

$$I_{|\psi_0\rangle} = I_{H|\psi_0\rangle} = H^n I_{|\psi_0\rangle} H^n,$$

so it's sufficient to implement $I_{|\psi_0\rangle}$. We claim that the following circuit works:



where T is the Toffoli gate construction implementing $|c\rangle |0\rangle_w |y\rangle_t \to |c\rangle |0\rangle_w |y \oplus c_1 \ldots c_n\rangle_t$, as above. To see that this works, note that for any $|x\rangle \neq |00..0\rangle$, $X^n |x\rangle$ contains at least one zero qubit, so T does nothing to $|-\rangle$ on the final line. Hence we just get back $|x\rangle$.

For $|x\rangle = |00 \ldots 0\rangle$, $X^n |x\rangle$ entirely comprises $|1\rangle$ qubits, so $|-\rangle \to -|-\rangle$ under the transformation. Thus $|00 \ldots 0\rangle \to -|00 \ldots 0\rangle$ and we're done.

The number of gates used in $2n + 2n + 2n - 2 + 1 = O(n)$ ($2n$ Hadamards, $2n$ NOTs, $2n-2$ Toffolis and 1 CX). □

4.5 Applications of ampl. ampl.

Example 1 (Grover with more than one good item): Suppose there are k good items in our Grover search. Then:

$$|\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{x \in B_n} |x\rangle$$

$$= \underbrace{\sqrt{\frac{k}{N}}}_{\sin(\theta)} \underbrace{\left(\frac{1}{\sqrt{k}} \sum_{\text{good } x\text{'s}} |x\rangle \right)}_{|g\rangle} + \underbrace{\sqrt{\frac{N-k}{N}}}_{\cos(\theta)} \underbrace{\left(\frac{1}{\sqrt{N-k}} \sum_{\text{bad } x\text{'s}} |x\rangle \right)}_{|b\rangle} .$$

In the context of ampl. ampl. then, we have

$$\sin(\theta) = \sqrt{\frac{k}{N}},$$

and that Q is a rotation through 2θ, where $\sin(\theta) = \sqrt{k/N} \Rightarrow \theta \approx \sqrt{k/N}$ for $k \ll N$.

So just apply the iteration operator Q, but this time do it

$$n \approx \frac{\pi}{4\theta} = \frac{\pi}{4} \sqrt{\frac{N}{k}}$$

times. Hence query complexity is $O(\sqrt{N/k})$. Note this example reduces to Grover's algorithm when $k = 1$.

Example 2 (Grover with unknown numbers): Consider Grover search with function $f : B_n \to B_1$, with k good x's, but k unknown. We then cannot apply standard Grover since the number of nudges needed (calculated in Example 1) depends on k. However, it's still possible to find a good x with good probability, as follows.

First, we prove the following geometric claim:

Lemma: Let the unit circle be populated with equally spaced points at angle γ apart from each other. Let l be any line through the centre. Then a fraction of at least

$$\frac{1}{2} - \frac{2\gamma}{\pi}$$

of the points are within 45° of l.

Proof: Best to draw a diagram. Within 45° of the line there are two sectors, each of angle $\pi/2$. There's a possibility these don't align with the γ angles, so the two γ regions at the sides of each sector might have their points missed out. Thus the number of points included in these regions is at least:

$$2 \cdot \left(\frac{\pi}{2\gamma} - 2 \right).$$

There are a total of $2\pi/\gamma$ points in the circle, so there's at least a fraction of

$$\frac{\gamma}{\pi} \left(\frac{\pi}{2\gamma} - 2 \right) = \frac{1}{2} - \frac{2\gamma}{\pi}$$

of the points covered, as required. $\square$

Now start with the uniform superposition $|\psi_0\rangle = \sin(\theta)|g\rangle + \cos(\theta)|b\rangle$, where

$$\sin(\theta) = \sqrt{\frac{k}{N}}, \quad \text{unknown.}$$

The Grover operator nudges things round by an angle $2\theta = 2\sqrt{k/N}$ at a time. In light of the Lemma, consider this to be our $\gamma = 2\sqrt{k/N}$. Then there are a total of

$$\frac{2\pi}{\gamma} = \frac{2\pi}{2\sqrt{k/N}} = \frac{\pi\sqrt{N}}{\sqrt{k}} < \pi\sqrt{N}$$

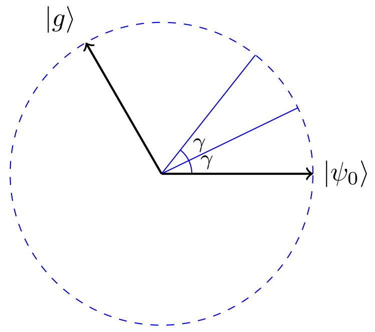
points at which our nudged state stops at round the circle. Choose to apply Q a total of K times then, with $0 < K < \pi\sqrt{N}$ chosen uniformly randomly. The chance we are in 45° of the good subspace is, by the Lemma,

$$\frac{1}{2} - \frac{2\gamma}{\pi} = \frac{1}{2} - \frac{4}{\pi}\sqrt{\frac{k}{N}} \approx \frac{1}{2},$$

for $N \gg k$. Clearly, if we are within 45° of the good subspace, then our probability of seeing a good element is at least $1/2$. Thus

$$\text{Prob (see good element)} \geq \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}.$$

Repeating a constant number of times thus gives success with any desired probability by the standard Probability Lemma. So the total number of queries to f is a constant times $O(\sqrt{N})$, which is just $O(\sqrt{N})$.



Example 3 (Square root speed-ups): Let A be a quantum algorithm (i.e. a sequence of unitary gates) with input state $|00...0\rangle$. The final state is $A|00...0\rangle$.

Consider the 'good states' the be the desired computational outcomes, and write $A|00...0\rangle = \alpha|a\rangle + \beta|b\rangle$, where $\alpha = \sin(\theta)$, and $|a\rangle$ is the normalised 'good' part of the outcome (in general this is an unequal superposition of good x). The probability of success of the algorithm is then $|\alpha|^2$, so we must repeat it $O(1/|\alpha|^2)$ times to see a good thing with any constant level of probability.

Instead, use ampl. ampl. Assuming we can check whether the answer is good or bad, we can implement

$$I_G|x\rangle = \begin{cases} -|x\rangle & x \text{ is good,} \\ |x\rangle & x \text{ is bad.} \end{cases}$$

Starting with state $|\psi\rangle = A|00...0\rangle$, the Grover operator is $Q = -I_{A|00...0\rangle}I_G = -(AI_{|00...0\rangle}A^\dagger)I_G$. Notice all parts are implementable: A is just the algorithm, $I_{|00...0\rangle}$ is implementable from above, and $A^\dagger$ is just the algorithm in reverse.

By the ampl. ampl. Theorem, Q is a rotation through 2θ such that $\sin(\theta) = |\alpha|$. Thus after

$$n \approx \frac{\pi}{4\theta} = O\left(\frac{1}{\theta}\right) = O\left(\frac{1}{\sin(\theta)}\right) = O\left(\frac{1}{|\alpha|}\right)$$

repetitions, $A|00...0\rangle$ will be rotated very near $|g\rangle$, and the final measurement will succeed with high probability.

Each application of Q needs one A and one $A^\dagger$, and notice that A and $A^\dagger$ clearly have the same time complexity. Thus $O(1/|\alpha|)$ repetitions of Q gives a square root speed-up over the direct method, which needs $O(1/|\alpha|^2)$ repeats.

Later, we'll see how ampl. ampl. can be made exact. Then the above has made a probabilistic algorithm into a deterministic one (a *derandomisation process*)!

Example 4 (Quantum counting): Suppose we have a function $f : B_n \to B_1$ with k 'good' x's. Let k be unknown. Then the task of *quantum counting* is to determine k, rather than to find some good x as we have previously done.

Recall that the matrix of Q is the $\{|b\rangle, |g\rangle\}$ basis is

$$\begin{pmatrix} \cos(2\theta) & -\sin(2\theta) \\ \sin(2\theta) & \cos(2\theta) \end{pmatrix}.$$

Its evectors are $|e_\pm\rangle = \frac{1}{\sqrt{2}}(|b\rangle \pm i\,|g\rangle)$ with corresponding evalues $\lambda_\pm = e^{\pm 2i\theta}$. By a short calculation, the uniform superposition can be written in terms of these evectors as:

$$|\psi_0\rangle = \frac{1}{\sqrt{2}}\left(e^{i\theta}|e_+\rangle + e^{-i\theta}|e_-\rangle\right).$$

Write $e^{\pm 2i\theta} = e^{2\pi i \phi_\pm}$, where $0 < \phi_\pm < 1$. Thinking about it, we must have

$$\phi_+ = \frac{2\theta}{2\pi} = \frac{\theta}{\pi} = \frac{1}{\pi}\sqrt{\frac{k}{N}},$$
$$\phi_- = \frac{-2\theta + 2\pi}{2\pi} = 1 - \frac{\theta}{\pi} = 1 - \frac{1}{\pi}\sqrt{\frac{k}{N}}.$$

Now run phase estimation on $U = Q$ with the eigenstate register set to $|\psi_0\rangle$. This will output an approximation to ϕ_+ or ϕ_- with equal probability $1/2$. For small θ, it's easy to tell which is which, as it will either be closed to 0 or 1.

Thus phase estimation gives, for m qubit lines, an m bit approximation to $\sqrt{k/N}$. Phase estimation uses 2^m controlled Q gates, so needs 2^m queries. Thus by the Phase Estimation Theorem we learn $\sqrt{k/N}$ to within an additive error $O(1/2^m$, with constant probability greater than $4/\pi^2$ using $O(2^m)$ queries to U_f.

Write $1/2^m = \delta/\sqrt{N}$ so that δ is the additive error in $\sqrt{k}$. Then the error in k is $O(\delta\sqrt{k})$ (from $\Delta(x^2) = 2x\Delta x$) using $O(\sqrt{N}/\delta)$ queries. It can be shown that this is the square of the number of operations needed to achieve the same error in the classical regime.

Example 5 (The unique collision problem): Let $f : B_n \to B_n$ be a one-to-one function on all inputs, except for a single pair x_1, x_2 with $f(x_1) = f(x_2)$. That is f has a *unique collision*. The *unique collision problem* is to determine both x_1 and x_2. Let Q be the query complexity of the optimal algorithm for the unique collision problem.

It's easy to see that $O(N) > Q$, since we can just check all values to find a collision.

We can get a lower bound as follows. Suppose that $g : B_n \to B_1$ is the function for a unique item Grover search problem. Then $f : B_n \to B_n$ defined by $f(x) = g(x) + x \pmod{2^n}$ has a unique collision at the 'good' x, say x_0, since $f(x_0) = g(x_0) + x_0 = x_0 + 1$, whilst $f(x) = x$ for all bad x. Thus $f(x_0) = f(x_0 + 1)$.

It follows that if we can solve the unique collision problem with time complexity Q, we can solve unique element search with time complexity Q. But Grover's algorithm is optimal for unique element search, so $O(\sqrt{N}) < Q$.

We now develop an algorithm that uses $O(N^{3/4})$ queries. Begin by partitioning B_n into random subsets A_k, each of size $\sqrt{N}$. Let k be 'good' if A_k contains *both* x_1 and x_2; assume for now that such an A_k exists (there can be at most one). Now define an indicator function $g : \{A_k\} \to \{0, 1\}$ by

$$g(A_k) = \begin{cases} 0 & \text{if } k \text{ is bad,} \\ 1 & \text{if } k \text{ is good.} \end{cases}$$

One evaluation of $g(A_k)$ can be achieved by just checking all the elements in A_k, which has f query complexity $O(\sqrt{N})$. If we perform a Grover search with g as our search function, we find a good A_k with g query complexity $O(\sqrt{\sqrt{N}}) = O(N^{1/4})$, since the number of A_k's is $\sqrt{N}$. Thus the total f query complexity is:

$$\underbrace{O(\sqrt{N})}_{\text{each use of } g} \cdot \underbrace{O(N^{1/4})}_{\text{number of uses of } g} = O(N^{3/4}).$$

Alas, it is unlikely that a good A_k will even exist, so our Grover search will probably fail.

If it does fail (it's easy to check - just compute $g(A_k)$ for the proposed 'good' A_k, taking only $O(\sqrt{N})$ time), we do the following. Now define A_k to be 'good' if it contains *exactly one* of x_1 and x_2 (so the other is in $B_n \backslash A_k$). Define a new indicator function:

$$h(A_k) = \begin{cases} 0 & \text{if } A_k \text{ is bad,} \\ 1 & \text{if } A_k \text{ is good.} \end{cases}$$

Can we implement $h(k)$ efficiently? Yes! To find $h(A_k)$, first evaluate $f(x)$ for all $x \in A_k$ and store the values. Denote this set of values by $f(A_k)$. Now perform a Grover search over the $N - \sqrt{N}$ values in $B_n \backslash A_k$, where an x is 'good' if it is on the list $f(A_k)$ (there can be only one such x!). The f query complexity of this is, by the general theory of Grover's algorithm,

$$O\left(\sqrt{N - \sqrt{N}}\right) = O(\sqrt{N}).$$

Note that evaluating $h(A_k)$ causes us to actually find x_1 and x_2. Now perform Grover search with indicator function h. There are two good items in size $\sqrt{N}$ search space, so we need $O(N^{1/4})$ queries to h to find a single good A_k. This has f query complexity $O(N^{1/4}) \cdot O(\sqrt{N}) = O(N^{3/4})$.

Now simply evaluate $h(A_k)$ using $O(\sqrt{N})$ queries to find x_1, x_2. The total query complexity is $O(N^{3/4})$.

The complete algorithm is therefore this: (1) perform the naïve method assuming only one good A_k; (2) if that fails, assume two good A_k and apply the more complex method; (3) if both fail, it's down to probability, and since we succeed with a constant probability, we can just repeat as many times as needed.

4.6 Making ampl. ampl. exact

Theorem: Amplitude amplification can be made exact with at most one extra query to the indicator function.

Proof: Let $|\psi\rangle = \alpha|g\rangle + \beta|b\rangle$ be the starting state for ampl. ampl., where $\sin(\theta) = \alpha$. We know the algorithm is exact if

$$n = \frac{\pi}{4\theta} - \frac{1}{2}$$

is an integer. Generally, though, n is not integral. To fix this, choose $\theta' < \theta$ such that

$$\lceil n \rceil = \frac{\pi}{4\theta'} - \frac{1}{2}.$$

Then if we started with the state $|\psi'\rangle = \alpha'|g\rangle + \beta'|b'\rangle$, with $\alpha' = \sin(\theta')$ (potentially redefining the bad subspace), we'd have an exact algorithm, applying Q $\lceil n \rceil$ times.

To achieve this, add a qubit to the system to make:

$$|\phi\rangle = \alpha|g\rangle|0\rangle + \beta|b\rangle|0\rangle .$$

Apply the following rotation to the qubit:

$$U = \begin{pmatrix} \alpha'/\alpha & -\sqrt{1-\alpha'^2/\alpha^2} \\ \sqrt{1-\alpha'^2/\alpha^2} & \alpha'/\alpha \end{pmatrix}.$$

we can make this because $\alpha' < \alpha$ and we know α, α'. Therefore: $I \otimes U|\phi\rangle =$

$$\alpha'|g\rangle|0\rangle + \alpha\sqrt{1-\frac{\alpha'^2}{\alpha^2}}|g\rangle|1\rangle + \frac{\beta}{\alpha'}\alpha|b\rangle|0\rangle + \beta\sqrt{1-\frac{\alpha'^2}{\alpha^2}}|b\rangle|1\rangle .$$

Redefine the good subspace to mean strings with 'good' first n bits, then a zero at the end. Then for our new state, $\alpha' = \sin(\theta')$, so ampl. ampl. will be exact. The number of rotations is $\lceil n \rceil$ which is at most one more than using the initial amplitude α. □

5 Hamiltonian simulation

5.1 Quantum dynamics

In a physical system with state vectors, there exists a self-adjoint operator H called the *Hamiltonian*. It represents energy. Recall that time-evolution of a quantum system is given by *Schrödinger's equation*

$$i\frac{d|\psi\rangle}{dt} = H|\psi\rangle .$$

We will consider only time-independent Hamiltonians, i.e. H does not depend on t. Then the formal solution of the Schrödinger equation is

$$|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle .$$

The matrix exponential is, as usual, defined by its power series and converges for all matrices. Thus to approximate quantum dynamics, we must approximate e^{-iHt}.

5.2 Operator norms

Approximation (closeness) of operators is given by the *operator/spectral norm*:

Definition: The *operator norm* of A is defined by

$$||A|| = \max_{|||\psi\rangle||=1} ||A|\psi\rangle||.$$

Theorem: We have

1. $||A|| =$ modulus of largest eigenvalue of A;
2. $||A+B|| \leq ||A|| + ||B||$;
3. $||AB|| \leq ||A|| \cdot ||B||$;
4. $||AU|| = ||UA|| = ||A||$ for unitary U.

Proof: Trivial. □

5.3 k-local Hamiltonians

Definition: Let H be a Hamiltonian acting on n qubits, i.e. a $2^n \times 2^n$ matrix. H is called *k-local* if it can be written as

$$H = \sum_{j=1}^{m} H_j,$$

where each H_j is a Hermitian matrix acting on at most k qubits (not necessarily next door to one another).

Theorem: In the above, $m = O(n^k)$.

Proof: We clearly have $m \leq \binom{n}{k}$. By Stirling's formula, $\binom{n}{k} = O(n^k)$ and the result follows. □

In particular, this means there are $O(n^k) \cdot O(2^k) = O(n^k)$ non-zero terms in the matrix H. This means we can actually read the data in it in polynomial time! (If it were $2^n \times 2^n$, this wouldn't be possible!)

Example 1: $H = X \otimes I \otimes I - 5Z \otimes I \otimes Y$ is 2-local on 3 qubits.

Example 2: Let $M_{(i,j)}$ denote the operator M acting on the ith and jth qubit and I on all the others. The *Ising model* then has Hamiltonian

$$H = J \sum_{i,j=1}^{n-1} Z_{(i,j)} Z_{(i,j+1)} + Z_{(i,j)} Z_{(i+1,j)}.$$

This is 3-local on n qubits.

This generalise to the *Heisenberg model*:

$$H = \sum_{i=1}^{n-1} J_X X_{(i)} X_{(i+1)} + J_y Y(i) Y_{(i+1)} + J_Z Z_{(i)} Z_{(i+1)}.$$

Example 3: Basically all of chemistry can be framed in terms of k-local Hamiltonians.

5.4 Commuting k-local Hamiltonians

We now try to simulate k-local Hamiltonians. This is in general hard because

$$\exp\left(-i \sum_j H_j t\right) \neq \prod_j \exp\left(-iH_j t\right)$$

if the H_j are non-commuting. We'll therefore begin by assuming that the H_j's *are* commuting, and we'll come back to the non-commuting case later.

In the case where the H_j's are commuting, we need only approximate

$$\exp\left(-iH_j t\right)$$

separately. We often want to do this in terms of a standard universal gate set; we can then use the *Solovay-Kitaev Theorem*:

Solovay-Kitaev Theorem: Let U be a unitary operator on k qubits and let S be a universal gate set. Then U can be approximated to within ϵ using $O(\log^c(1/\epsilon))$ gates from S with $c < 4$.

Proof: Not required. □

Theorem: Let $\{U_i\}$ and $\{V_i\}$ be sets of unitary operators with $||U_i - V_i|| \leq \epsilon$ for all $i = 1, 2..., m$. Then $||U_1...U_m - V_1...V_m|| \leq m\epsilon$.

Proof: In the $n = 2$ case, we have

$$\begin{aligned} ||U_2U_1 - V_2V_1|| &= ||(U_2 - V_2)U_1 + V_2(U_1 - V_1)|| \\ &\leq ||U_2 - V_2|| + ||U_1 - V_1|| = 2\epsilon. \end{aligned}$$

Then carry on by induction. □

Using these results, it's clear what to do:

Theorem: For H_j commuting, we can approximate e^{-iHt} to within ϵ by a circuit of size $O(n^k \text{poly}(\log(n^k/\epsilon)))$, with gates chosen from any universal gate set.

Note that since $m = O(n^k)$, this circuit size is polynomial in both n and $\log(1/\epsilon)$. Note also that $\log(1/\epsilon)$ is the number of digits of precision in the approximation.

Proof: Since the H_j's commute, we have

$$e^{-iHt} = \prod_{j=1}^{m} e^{-iH_j t}.$$

By the Solovay-Kitaev Theorem, each $e^{-iH_j t}$ can be approximated to within ϵ/m with $O(\text{poly}(\log(m/\epsilon)))$ gates from any universal gate set. Then from the second Theorem on error accumulation, we have that the full product can be approximated to within $m(\epsilon/m) = \epsilon$ using

$$O(m\text{poly}(\log(m/\epsilon))) = O(n^k\text{poly}(\log(n^k/\epsilon))) \quad \text{gates.}$$ □

5.5 Non-commuting case

Definition: For any matrix X, write $X + O(\epsilon)$ for $X + E$ where E is a matrix with norm $||E|| = O(\epsilon)$.

Theorem (Lie-Trotter Product Formula): Let A and B be matrices with $||A|| \leq \kappa$ and $||B|| \leq \kappa$, for $\kappa \ll 1$. Then

$$e^{-iA}e^{-iB} = e^{-i(A+B)} + O(\kappa^2).$$

Proof: We have

$$e^{-iA} = I - iA + (iA)^2 \sum_{k=0}^{\infty} \frac{(-iA)^k}{(k+2)!}.$$

The norm of the sum can be bounded by:

$$\left|\left| \sum_{k=0}^{\infty} \frac{(-iA)^k}{(k+2)!} \right|\right| \leq \sum_{k=0}^{\infty} \frac{||A||^k}{k!} \leq e^{-\kappa} < 1,$$

in particular using $(k+2)! > k!$. Therefore, noting that $||(-iA)^2|| \leq \kappa^2$, we have

$$e^{-iA} = I - iA + O(\kappa^2).$$

Therefore, we have

$$e^{-iA}e^{-iB} = (I - iA + O(\kappa^2))(I - iB + O(\kappa^2)) = I - i(A+B) + O(\kappa^2).$$

Now since $||A + B|| \leq ||A|| + ||B|| = 2\kappa = O(\kappa)$, we have $e^{-i(A+B)} = I - i(A + B) + O(\kappa^2)$. The formula follows. □

The Lie-Trotter product formula lets us get around the non-commuting problem by essentially ignoring it.

Theorem: In the non-commuting case, e^{-iHt} can be approximated to within ϵ by a circuit of size

$$O\left(\frac{n^{4k}t^2}{\epsilon}\text{poly}\left(\log\left(\frac{n^{4k}t^2}{\epsilon^2}\right)\right)\right),$$

with gates chosen from any universal gate set.

Proof: In general, $||H_j||$ can be large, which is problematic when we want to use Lie-Trotter. To remedy this, choose κ such that $||H_j|| < \kappa$ for all j (this may be large) and introduce N large so that

$$||H_j\frac{t}{N}|| < \frac{\kappa t}{N}.$$

Therefore H_jt/N have small norms (essentially, we have split up the time evolution into small t/N sized steps). Now consider:

$$U = e^{-i(H_1+...+H_m)t} = \left(\exp\left(-i\left(\frac{H_1t}{N}+...+\frac{H_mt}{N}\right)\right)\right)^N$$

We note this is just the operator $\exp\left(-i\left(\frac{H_1t}{N}+...+\frac{H_mt}{N}\right)\right)$ applied N times. We want the final error in U to be less than ϵ, hence we want the error in

$$\exp\left(-i\left(\frac{H_1t}{N}+...+\frac{H_mt}{N}\right)\right)$$

to be less than ϵ/N, by the error propagation Theorem.

Let's now work out our approximation for this operator. Since H_jt/N has small norm, we can use Lie-Trotter:

$$e^{-iH_1t/N}e^{-iH_2t/N}...e^{-iH_mt/N}$$
$$= \left(e^{-i(H_1+H_2)t/N} + O\left(\frac{\kappa^2t^2}{N^2}\right)\right)e^{-iH_3t/N}...e^{-iH_mt/N},$$

by Lie-Trotter. Notice that $||(H_1+H_2)t/N|| \leq 2\kappa t/N$. So we can reapply Lie-Trotter to include the next term, and the next, and the next, etc. until we get

$$e^{-iH_1t/N}...e^{-iH_mt/N} = e^{-i(H_1+...+H_m)t/N}$$
$$+O\left(\frac{\kappa^2t^2}{N^2}\right) + O\left(\frac{(2\kappa)^2t^2}{N^2}\right) + ... + O\left(\frac{((m-1)\kappa)^2t^2}{N^2}\right).$$

Recalling that the sum of squares is a cubic, we have:

$$e^{-iH_1t/N}...e^{-iH_mt/N} = e^{-i(H_1+...+H_m)t/N} + O\left(\frac{m^3\kappa^2t^2}{N^2}\right).$$

Thus the requirement on ϵ/N is, for some constant C:

$$C\frac{m^3\kappa^2t^2}{N^2} < \frac{\epsilon}{N} \quad\Rightarrow\quad N > \frac{Cm^3\kappa^2t^2}{\epsilon}.$$

Thus we need $N = O(m^3\kappa^2t^2/\epsilon)$ to achieve error ϵ.

Naïvely, we assume that $e^{-iH_jt/N}$ can all be implemented exactly. Then the circuit size is

$$O(Nm) = O\left(\frac{m^4\kappa^2t^2}{\epsilon}\right) = O\left(\frac{n^{4k}t^2}{\epsilon}\right),$$

since κ is constant.

The circuit we've made consists only of $e^{-iH_jt/N}$ operations. To incorporate an approximation to $e^{-iH_jt/N}$, we recall that the Solovay-Kitaev Theorem says we can approximate $e^{-iH_jt/N}$ to within order $O(\epsilon/C)$ by a circuit of size $O(\text{poly}(\log(C/\epsilon)))$ for whatever C we choose.

By the error propagation Theorem, we must choose

$$C = O\left(\frac{n^{4k}t^2}{\epsilon}\right),$$

the size of our naïve circuit for the overall error to remain ϵ. Therefore the number of gates we need to use is:

$$O\left(\frac{n^{4k}t^2}{\epsilon}\right)\cdot O\left(\text{poly}\left(\log\left(\frac{n^{4k}t^2}{\epsilon^2}\right)\right)\right)$$
$$= O\left(\frac{n^{4k}t^2}{\epsilon}\text{poly}\left(\log\left(\frac{n^{4k}t^2}{\epsilon^2}\right)\right)\right). \quad \square$$

Remark: For fixed n, ϵ, our algorithm takes time $O(t^2)$ to complete. In real life, the quantum system runs for time t. This occurs because of our use of the Lie-Trotter product formula.

In fact, better 'splitting formula' exist to approximate e^{A+B} (we'll see *Strang splitting*, for example, below). In general it's possible to show that the time for our quantum algorithm can be improved to $O(t^{\delta+1})$ for any $\delta > 0$.

5.6 Strang splitting

Theorem: For $||A|| \leq \kappa$ and $||B|| \leq \kappa$, with $\kappa \ll 1$, we have the *Strang splitting*:

$$e^{-iA/2}e^{-iB}e^{-iA/2} = e^{-i(A+B)} + O(\kappa^3).$$

Proof: Just as for Lie-Trotter, we have $e^{-iA/2}e^{-iB}e^{-iA/2} =$

$$\left(I - \frac{iA}{2} - \frac{A^2}{8} + O(\kappa^3)\right)\left(I - iB - \frac{B^2}{2} + O(\kappa^3)\right)$$
$$\cdot\left(I - \frac{iA}{2} - \frac{A^2}{8} + O(\kappa^3)\right).$$

Multiplying everything out, we find the result. $\square$

Theorem: In the non-commuting case, e^{-iHt} can be approximated to within ϵ by a circuit of size

$$O\left(\frac{n^{3k}t^{3/2}}{\epsilon^{1/2}}\text{poly}\left(\log\left(\frac{n^{3k}t^{3/2}}{\epsilon^{3/2}}\right)\right)\right),$$

with gates chosen from any universal gate set.

Proof: Everything's the same up until when we work out the approximation to $e^{-iH_1t/N}...e^{-iH_mt/N}$. At this point, instead of using Lie-Trotter, we use Strang splitting. We have:

$$e^{-i(H_1+H_2)t/N} = e^{-iH_2t/2N}e^{-iH_1t/N}e^{-iH_2t/2N} + O\left(\frac{\kappa^3 t^3}{N^3}\right).$$

Since $||(H_1+H_2)t/N|| \leq 2\kappa t/N$, we can iterate just as in the Lie-Trotter case to find: $e^{-i(H_1+...+H_m)t/N} =$

$$e^{-iH_mt/2N}...e^{-iH_2t/2N}e^{-iH_1t/N}e^{-iH_1t/2N}...e^{-iH_mt/2N}$$
$$+O\left(\frac{\kappa^3t^3}{N^3}\right) + O\left(\frac{(2\kappa)^3t^3}{N^3}\right) + ... + O\left(\frac{((m-1)^3\kappa^3t^3}{N^3}\right).$$

Recall that the sum of cubes is a fourth power, so the error in the Strang splitting is

$$O\left(\frac{m^4\kappa^3t^3}{N^3}\right) = O\left(\frac{m^4t^3}{N^3}\right).$$

For this to be less than ϵ/N, we need

$$N = O\left(\frac{m^2t^{3/2}}{\epsilon^{1/2}}\right).$$

Then, in the naïve circuit, we need (from the Strang splitting) $N(2m+1) = O(mN)$ operators. Thus the naïve circuit size is

$$O\left(\frac{m^3t^{3/2}}{\epsilon^{1/2}}\right) = O\left(\frac{n^{3k}t^{3/2}}{\epsilon^{1/2}}\right).$$

Now use Solovay-Kitaev argument to get result. □

5.7 Diagonalisable Hamiltonians

When Hamiltonians are diagonalisable by unitary matrices that can be implemented efficiently, Hamiltonian simulation becomes much easier:

Theorem: Let H be a Hamiltonian that can be diagonalised as $H = UDU^\dagger$, for U unitary, and implementable by an $O(\text{poly}(n))$ size circuit. D is a diagonal matrix:

$$D = \sum_x |x\rangle\langle x|.$$

Suppose that $|x\rangle \mapsto e^{-id(x)t}|x\rangle$ can be implemented in $O(\text{poly}(n))$ time. Then e^{-iHt} can be implemented in $O(\text{poly}(n))$ time.

Proof: Simply note that $H^k = (UDU^\dagger)^\dagger = UD^kU^\dagger$, so

$$e^{-iHt} = Ue^{-iDt}U^\dagger.$$

Now $e^{-iDt}|x\rangle = e^{-id(x)t}|x\rangle$ is efficiently implementable, by assumption, and so are U, $U^\dagger$. So we're done. □

6 Harrow-Hassidim-Lloyd algorithm

6.1 Problem setup

We want to solve the linear system $A\mathbf{x} = \mathbf{b}$, where $\mathbf{k}, \mathbf{x} \in \mathbb{C}^N$ and N is so large that Gaussian elimination is inefficient. Assume $N = 2^n$ (we can always add zero equations to get to a power of 2).

The best known classical algorithms take $O(\text{poly}(N))$ time, but we'll develop an algorithm that runs in $O(\text{poly}(\log(N))) = O(\text{poly}(n))$ time.

Remarks: (1) Writing the answer down takes $O(N)$ time! So we'll want to compute *properties* of the solution, rather than the solution itself, e.g. quadratic expressions like $\mathbf{x}^TM\mathbf{x}$ (if M were diagonal with some 1's and 0's, we could compute any particular x_i we desired).

(2) It takes $O(N^2)$ time just to read A, and $O(N)$ time to read b! So we'll have to assume a special form of A, b in order to make an improvement.

We will need the following important parameters:

- The system size N.
- The desired approximation tolerance, ϵ.
- The *condition number* of the matrix A. This is the ratio of the modulus of the largest to the modulus of the smallest eigenvalue:
$$\kappa = \frac{|\lambda_{\text{max}}|}{|\lambda_{\text{min}}|}.$$
In classical numerical analysis, having κ close to ∞ means the approximation will be bad. This is because κ provides a measure of how close A is to being non-invertible; this can be seen by renormalising A so that $|\lambda_{\text{max}}| = 1$, for then
$$|\lambda_{\text{min}}| = \frac{1}{\kappa}.$$
Then as $\kappa \to \infty$, $|\lambda_{\text{min}}| \to 0$, implying A becomes non-invertible at $\kappa = \infty$.
- A property of the solution we would like to calculate, given as a quadratic form $\mu = \mathbf{x}^TM\mathbf{x}$.

6.2 Assumptions on A, b and μ

As stated above, we can't even read A in poly(n) time. Therefore, we make the following restrictions on A:

1. A is Hermitian. This does not constitute a loss of generality, since if it is not, we can redefine the system as:
$$\begin{pmatrix} 0 & A^\dagger \\ A & 0 \end{pmatrix}\begin{pmatrix} \mathbf{x} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{b} \end{pmatrix},$$
merely doubling its enormous size.

2. We require the condition number κ to be suitably small; in particular, we'll assume κ is bounded by poly(n). Such matrices are called *well-conditioned*.

   We will also assume that A has been scaled so that $\lambda_{\max} = 1$ (note evalues are real, since Hermitian). Then all of A's eigenvalues are in $[1/\kappa, 1] = [1/\text{poly}(n), 1]$, i.e. they're all close to 1.

3. The algorithm uses Hamiltonian simulation. So we need e^{iAt_0} to be implementable in $O(\text{poly}(n, t_0))$ time.

   Any class of matrices that satisfies this condition will work in the HHL algorithm. We've already seen that k-local A would do. However, these are not the most general form; we'll expand our scope by considering matrices that are *row-sparse* and *row-computable*. These are the properties we'll assume of A.

Definition: A matrix A is *row-sparse* if each row contains at most poly(n) non-zero entries. More generally, a matrix is s-sparse if each row contains at most s non-zero entries. Note an row-sparse matrix is an s-sparse matrix with $s = O(\text{poly}(N))$.

Definition: An s-sparse matrix A is *row-computable* if the entries of A can be computed in the following sense: there is a classical $O(s)$-time algorithm C which, given any row index $1 \leq i \leq N$ and integer k, output the kth non-zero entry A_{ij} or row i and its column location j. That is, $C(i, k) = (j, A_{ij})$.

Theorem: Let A an s-sparse and row-computable. The operator e^{iAt_0} can be implemented up to error ϵ, by a quantum circuit of size $O(\log(N)s^2t_0)$.

Proof: Not required. □

For **b**, we assume the following:

4. **b** has unit length. This does not constitute a loss of generality, since if we work with the normalised $\hat{\mathbf{b}}$ instead, we can just rescale the answer by $|\mathbf{b}|$ at the end:
$$A\frac{\mathbf{x}}{|\mathbf{b}|} = \hat{\mathbf{b}}.$$

5. The $\log(N)$-qubit state
$$\sum_{i=0}^{2^n-1} b_i \left|i\right\rangle$$
can be efficiently produced on a quantum computer in time $O(\text{poly}(n))$. We'll discuss this in more detail later.

Finally, for μ we assume:

6. $\mu = \mathbf{x}^T M\mathbf{x}$ has M Hermitian. This does not constitute a loss of generality, since if M is not Hermitian, we can write $M = K + iL$, where
$$K = \frac{1}{2}(M + M^\dagger), \quad L = \frac{1}{2i}(M - M^\dagger),$$
so that K and L are both Hermitian. We can then do the algorithm for K and L separately to obtain $\mathbf{x}^T L\mathbf{X}$, $\mathbf{x}^T K\mathbf{x}$, and then combine at the end.

7. Notice μ is the expectation value of the observable M (assuming it is Hermitian). Thus we assume the measurement corresponding to M can be done in poly$(\log(N))$ time.

With all these assumptions in place, the best known classical algorithm runs in $O(Ns\sqrt{\kappa}\log(1/\epsilon))$ time. The HHL algorithm runs in
$$O(\log(N)s^2\kappa^2/\epsilon)$$
time, giving the solution to within ϵ.

In the regime where $\epsilon = 1/\text{poly}(\log(N))$, and A is well-conditioned ($\kappa = O(\text{poly}(\log(N)))$) and row-sparse ($s = O(\text{poly}(\log(N)))$), then the classical complexity is $O(\text{poly}(N))$, whilst the quantum complexity is $O(\text{poly}(\log(N)))$. Thus HHL constitutes an *exponential speed up*.

6.3 The algorithm

For clarity of exposition, we'll assume that phase estimation and Hamiltonian simulation work perfectly. The error analysis of HHL turns out to be very ugly and is non-examinable. The algorithm is as follows:

The HHL algorithm: Work in an N-dimensional space with basis $\{\left|i\right\rangle : i = 0, 1..., N-1\}$, i.e. $\log(N)$ qubits. Let the eigenvectors of A be $\left|u_j\right\rangle$ with corresponding eigenvalues λ_j, for $j = 0, 1, ..., N-1$.

1. Begin the algorithm by implementing the RHS as a quantum state $\left|\mathbf{b}\right\rangle$, given by
$$\left|\mathbf{b}\right\rangle = \sum_{i=0}^{N-1} b_i \left|i\right\rangle = \sum_{j=0}^{N_1} \beta_j \left|u_j\right\rangle.$$

   We assume this is possible here, but will prove that we can do it later in the course.

   Then our desired state is
$$\left|x\right\rangle = A^{-1}\left|b\right\rangle = \sum_{j=0}^{N-1} \frac{\beta_j}{\lambda_j} \left|u_j\right\rangle.$$

   It will be the object of the rest of the algorithm to compute this. Note this is well-defined since by assumption the evalues are bounded well away from zero.

Clearly the operation of A^{-1} on $|b\rangle$ is linear, but it is *not unitary*. Therefore we need to use phase estimation, which can implement a non-unitary operator.

2. Apply the unitary phase estimation operator U_{PE} for the unitary operator $e^{2\pi i A}$. The exponential here, and its controlled powers, are implemented by Hamiltonian simulation.

   The operator $e^{2\pi i A}$ has eigenvalues $e^{2\pi i \lambda_j}$. Therefore phase estimation gives:

$$U_{\text{PE}} |b\rangle |0\rangle \sum_{j=0}^{N-1} \beta_j |u_j\rangle |\lambda_j\rangle .$$

3. Adjoin an ancilla qubit in state $|0\rangle$. Consider the *controlled rotation* $C\text{-}ROT$:

$$C\text{-}ROT\, |x\rangle |0\rangle = \sqrt{1 - \frac{c^2}{x^2}}\, |x\rangle |0\rangle + \frac{c}{x} |x\rangle |1\rangle .$$

   Notice that C- is just some fixed operation on $n+1$ qubits. It does not depend on the system $A\mathbf{x} = \mathbf{b}$ at all. We'll discuss how to implement such a rotation later in the course.

   Apply $C\text{-}ROT$ to the state from above, controlled on the $|\lambda_j\rangle$ states. Then we get:

$$\sum_{j=0}^{N-1} \beta_j \sqrt{1 - \frac{c^2}{\lambda_j^2}}\, |u_j\rangle |\lambda_j\rangle |0\rangle + \frac{\beta_j c}{\lambda_j} |u_j\rangle |\lambda_j\rangle |1\rangle .$$

   We want the part associated to $|1\rangle$.

   Note here that c must be chosen in $C\text{-}ROT$ such that $c \leq |\lambda_{\text{min}}|$. This is to ensure $|c/\lambda_j| < 1$, so that the amplitudes here don't exceed 1. For definiteness, one might take $c = 1/\kappa$ for example.

4. We use *post-selection*. Measure the ancilla qubit, hoping to get result 1. We see 1 with probability:

$$P = \left|\left| \sum_{j=0}^{N-1} \frac{\beta_j c}{\lambda_j} |u_j\rangle |\lambda_j\rangle \right|\right|^2 = \sum_{j=0}^{N-1} \frac{|\beta_j|^2 |c|^2}{|\lambda_j|^2}$$
$$= \frac{1}{\kappa^2} \sum_{j=0}^{N-1} \frac{|\beta_j|^2}{|\lambda_j|^2} \geq \frac{1}{\kappa^2}.$$

   This follows because

$$\sum_{j=0}^{N-1} |\beta_j|^2 = 1, \qquad \frac{1}{|\lambda_j|^2} \geq 1.$$

   So we get 1 with constant probability, and therefore can afford to keep repeating the algorithm $O(\kappa^2)$ times until we get 1.

   As an alternative, one might use amplitude amplification. The $O(\kappa^2)$ repetitions can be reduced to $O(\kappa)$ repetitions. Here, the 'good subspace' consists of anything with a $|1\rangle$ in the final register.

5. We are left with the state

$$|\hat{y}\rangle = \frac{c}{\sqrt{P}} \sum_{j=0}^{N-1} \frac{\beta_j}{\lambda_j} |u_j\rangle |\lambda_j\rangle .$$

   We need to erase the λ_j's. As usual, run phase estimation in reverse:

$$U_{\text{PE}}^{-1} |\hat{y}\rangle = \frac{c}{\sqrt{P}} \sum_{j=0}^{N-1} \frac{\beta_j}{\lambda_j} |u_j\rangle |0\rangle = \frac{c}{\sqrt{P}} |x\rangle |0\rangle .$$

   Discard the $|0\rangle$. The answer we obtain is

$$|\hat{x}\rangle = \frac{c}{\sqrt{P}} |x\rangle ,$$

   which is the normalised version of the solution.

6. To obtain the property μ, we perform measurements of the observable M on $|\hat{x}\rangle$ to estimate its mean value:

$$\hat{\mu} = \frac{c^2}{P} \langle x|M|x\rangle = \frac{c^2 \mu}{P}.$$

   Here, $\hat{\mu}$ is the normalised form of μ.

   According to the *Chernoff-Hoeffding bound*,

$$O\left(\frac{\log(1/\eta)}{\xi^2}\right)$$

   suffice to estimate the mean $\hat{\mu} = \langle \hat{x}|M|\hat{x}\rangle$ with any desired probability of success $1-\eta$, to any desired accuracy ξ.

7. We know what $c = 1/\kappa$ is, as we chose it when applying $C\text{-}ROT$. Similarly, we can estimate P in the post-selection step by applying the Chernoff-Hoeffding bound to the ancilla measurement outcome, whose mean is

$$0 \cdot p_0 + 1 \cdot p_1 = P,$$

   where p_0 is the probability of 0, $p_1 = P$ is the probability of 1. Using these pieces of information, we can find $\mu = \langle x|M|x\rangle$, the un-normalised form of μ.

6.4 Production of the state $|b\rangle$

In the HHL algorithm discussion above, we assumed we could make the state $|b\rangle$ efficiently. We will now show how, via an iterative procedure.

Theorem: Assuming that

$$g(k_1, k_2) = \sum_{i=k_1}^{k_2} b_i^2$$

can be made classically in $O(\text{poly}(n))$ time for any $0 \leq k_1 \leq k_2 \leq N-1$, we can make $|b\rangle$ in $O(\text{poly}(n))$ time.

Proof: Intuition: Let $a, b \geq 0$ be real with $a^2 + b^2 = 1$. Notice that

$$\begin{pmatrix} a & -b \\ b & a \end{pmatrix}$$

is unitary and maps $|0\rangle$ to $|\xi\rangle = a|0\rangle + b|1\rangle$.

Now suppose $s, t, u, v \geq 0$ and $s^2 + t^2 = a^2$, $u^2 + v^2 = b^2$. We now want to make the state

$$|\eta\rangle = s|00\rangle + t|01\rangle + u|10\rangle + v|11\rangle .$$

Begin by making $|\xi\rangle|0\rangle = a|0\rangle|0\rangle + b|1\rangle|0\rangle$. Now apply the controlled rotation

$$C\text{-}ROT\,|x\rangle|0\rangle = \cos(\theta_x)|x\rangle|0\rangle + \sin(\theta_x)|x\rangle|1\rangle .$$

This gives

$$|\eta\rangle = a\cos(\theta_0)|00\rangle + a\sin(\theta_0)|01\rangle + b\cos(\theta_1)|10\rangle + b\sin(\theta_1)|11\rangle .$$

This suggests we should pick $\cos(\theta_0) = s/a$ (if $a = 0$, the whole construction is easy, and we don't need controlled rotations). Then $\sin(\theta_0) = \sqrt{1 - s^2/a^2} = t/a$. We should also pick $\cos(\theta_1) = u/b$, so that $\sin(\theta_1) = \sqrt{1 - u^2/b^2} = v/b$. Thus we're left with:

$$|\eta\rangle = s|00\rangle + t|01\rangle + u|10\rangle + v|11\rangle .$$

Induction setup: We now iterate this process, by adjoining a $|0\rangle$ each time and applying a controlled rotation, now controlled by all previously existing qubits.

Let $B_{i_1 \ldots i_k}$ denote the set of bit strings in B_n constructed as follows. If $i_1 = 0$, delete the second half of B_n, and if $i_1 = 1$, delete the first half of B_n.

Now consider the remaining strings, if $i_2 = 0$, delete the second half of the remaining strings, and if $i_1 = 1$, delete the first half of the remaining strings. Iterate to get $B_{i_1 \ldots i_k}$.

Define $T_{i_1 \ldots i_k} = \sqrt{\sum_{x \in B_{i_1 \ldots i_k}} b_x^2}$, and $T_\emptyset = 1$ (where $\emptyset$ is the empty set). In particular, it's clear that $T_{i_1 \ldots i_n} = b_{i_1 \ldots i_n}$, and for any fixed k, we have

$$\sum_{i_1, i_2 \ldots i_k} T^2_{i_1 \ldots i_k} = \sum_x b_x^2 = 1.$$

Notice also that by assumption, we can make any of the $T_{i_1 \ldots i_k}$ in $O(\text{poly}(n))$ time.

For each $B_{i_1 \ldots i_k}$ introduce an associated angle $\theta_{i_1 \ldots i_k}$ defined by

$$\cos(\theta_{i_1 \ldots i_k}) = \frac{T_{i_1 \ldots i_k 0}}{T_{i_1 \ldots i_k}}, \quad \sin(\theta_{i_1 \ldots i_k}) = \frac{T_{i_1 \ldots i_k 1}}{T_{i_1 \ldots i_k}}.$$

Then the rotation $ROT(\theta_{i_1 \ldots i_k})$ maps $|0\rangle$ to

$$ROT(\theta_{i_1 \ldots i_k})|0\rangle = \frac{1}{T_{i_1 \ldots i_k}} \left(T_{i_1 \ldots i_k 0}|0\rangle + T_{i_1 \ldots i_k 1}|1\rangle\right)$$

Induction proof: Now begin the main proof. The base case begins with

$$1 = \sum_{x \in B} b_x^2 = T_\emptyset^2 .$$

Compute T_0, and hence compute $\cos(\theta_\emptyset) = T_0/1 = T_0$. Apply $ROT(\theta_\emptyset)$ to $|0\rangle$ to get $T_0|0\rangle + T_1|1\rangle$.

We now outline the induction step. Suppose we have made

$$|\psi_k\rangle = \sum_{i_1 \ldots i_k} T_{i_1 \ldots i_k} |i_1 \ldots i_k\rangle .$$

Adjoint a $|0\rangle$ qubit, and then apply the controlled rotation $C\text{-}ROT$, controlled on the $|i_1 \ldots i_k\rangle$ register. By the definition of the angle $\theta_{i_1 \ldots i_k}$, this gives

$$|\psi_{k+1}\rangle = \sum_{i_1 \ldots i_{k+1}} T_{i_1 \ldots i_{k+1}} |i_1 \ldots i_{k+1}\rangle .$$

After n steps, we get the desired state $|b\rangle$, as $T_{i_1 \ldots i_n} = b_{i_1 \ldots i_n}$. This is all $O(\text{poly}(n))$ since the controlled rotations can be implemented in $O(\text{poly}(n))$ time (see below). □

6.5 Controlled rotations

Theorem: Let $m = \text{poly}(\log(N)) = \text{poly}(n)$. Provided that the angle $\theta_x \in [0, \pi/2)$ can be computed from x in $O(m) = O(\text{poly}(n))$ time, the controlled rotation

$$C\text{-}ROT\,|x\rangle|0\rangle \to |x\rangle(\cos(\theta_x)|0\rangle + \sin(\theta_x)|1\rangle),$$

where $|x\rangle$ is an $m+1$ qubit register, can be implemented as an $O(m) = O(\text{poly}(n))$ size circuit using only 1 and 2 qubit gates.

Proof: Adjoin $m+1$ ancilla qubits in state $|0\rangle$ in the middle first, and put θ_x in them (using the $O(\text{poly}(n))$ time calculation of θ_x):

$$|x\rangle|0\rangle_0 \ldots |0\rangle_m |0\rangle \to |x\rangle|\theta_x\rangle|0\rangle .$$

Write $\theta_x = i_0.i_1 \ldots i_m$ in binary (assuming $\theta_x \in [0, \pi/2)$). Then we have:

$$R(\theta_x) = R(i_0) R\left(\frac{i_1}{2}\right) \ldots R\left(\frac{i_m}{2^m}\right).$$

Labelling the final qubit as t for target, we implement $C\text{-}ROT$ via the two qubit gates:

$$C\text{-}ROT(i_0)_{0t}\, C\text{-}ROT\left(\frac{i_1}{2}\right)_{1t} \ldots C\text{-}ROT\left(\frac{i_m}{2^m}\right)_{mt} .$$

Finally, just erase the $|\theta_x\rangle$ register by running the computation of θ_x backwards. Discard these qubits, and we're done. □

(In practice, θ_x will not be exact to $m+1$ bits, however this can be accounted for.)

6.6 Applications of HHL

Example (Dynamical systems): Let $x_t \in \mathbb{R}^N$ be the state vector of a discrete linear dynamical system with evolution rule:

$$x_{t+1} = \mathcal{L}x_t = Ax_t + b,$$

where A is an $N \times N$ matrix which is row-sparse and row computable and b is constant. Suppose that A is Hermitian and $||A|| \leq 1/2$, and that the system has been scaled to have $||b|| = 1$. A *stable state* is a state $s \in \mathbb{R}^N$ for which $\mathcal{L}s = s$.

Suppose we are given two such systems, (A, b) and (A', b'), whose stable states s and s' are either (i) within $\pi/6$ of each other; (ii) further than $\pi/3$ from one another. We can use HHL to decide which is which.

Note that the stable states of the system obey the linear system $(I - A)s = b$. Since $||A|| \leq 1/2$, the eigenvalues of $I - a$ lie in $[\frac{1}{2}, \frac{3}{2}]$, and so are bounded away from zero. Thus $I - A$ is well-conditioned. Since A is row-sparse and row-computable it's clear that $I - A$ is too. We assume $|b\rangle$ can be implemented in poly$(\log(N))$ time.

Therefore, HHL applied to $(I - A)s = b$ and $(I - A')s' = b'$ gives the states $|s\rangle$ and $|s'\rangle$, within tolerance ϵ of the real solution, in time $O($poly$(\log(N))/\epsilon)$ (we haven't seen this explicitly, but it was stated before the outline of the HHL algorithm).

The given conditions mean we have either:

$$|\langle s|s'\rangle|^2 \geq \cos^2\left(\frac{\pi}{6}\right) = \frac{3}{4}, \text{ or } |\langle s|s'\rangle|^2 \leq \cos^2\left(\frac{\pi}{3}\right) \leq \frac{1}{4}.$$

Consider applying the swap test to $|s\rangle$ and $|s'\rangle$; then we have

$$\text{Prob}(0) = \frac{1 + |\langle s|s'\rangle|^2}{2},$$

as before. In the *ideal* case, when $|s\rangle$ and $|s'\rangle$ are *exact*, we have

$$\text{Prob}(0) \geq \frac{7}{8}, \quad \text{or} \quad \text{Prob}(0) \leq \frac{5}{8}.$$

So we can distinguish the states by computing these probabilities (which will involve use of the Chernoff-Hoeffding bound).

However, the states are *not* exact. If $|\tilde{s}\rangle$ and $|\tilde{s}'\rangle$ are the true solutions, we have

$$||\,|\tilde{s}\rangle - |s\rangle\,|| < \epsilon, \quad ||\,|\tilde{s}'\rangle - |s'\rangle\,|| < \epsilon.$$

This implies that (we'll assume this here):

$$\Big|\, |\langle \tilde{s}|\tilde{s}'\rangle| - |\langle s|s'\rangle| \,\Big| < f(\epsilon),$$

for some $f(\epsilon)$ which obeys $f(\epsilon) \to 0$ as $\epsilon \to 0$.

Therefore for any small $\eta > 0$, we can, with suitable corresponding small ϵ, have

$$\text{Prob}(0) \geq \frac{7}{8} - \eta, \quad \text{or} \quad \text{Prob}(0) \leq \frac{5}{8} + \eta.$$

The probabilities are separated by $\frac{1}{4} + 2\eta$. Therefore a good estimate of the probabilities will let us decide which case is which.

For this, we use the *Chernoff-Hoeffding bound*: if we estimate Prob(0) as frequency f of 0 seen in k samplings of a $0/1$ distribution, then

$$|f - \text{Prob}(0)| < \xi \text{ with probability } \geq 1 - \epsilon,$$

if $k \geq \log(1/\epsilon)/2\xi^2$.

Thus we need only sample the swap test $O(\log(1/\epsilon))$ times for any η. So the whole process runs in time:

$$O\left(\text{poly}(\log(N)) \cdot \frac{1}{\epsilon}\log\left(\frac{1}{\epsilon}\right)\right) = O\left(\text{poly}(\log(N))\right),$$

for fixed tolerance ϵ.