Fast Black-Box Quantum State Preparation

Johannes Bausch

May 2021

Quantum state preparation is an important ingredient for other higher-level quantum algorithms, such as Hamiltonian simulation, or for loading distributions into a quantum device to be used e.g. in the context of optimization tasks such as machine learning. Starting with a generic "black box" method devised by Grover, which employs amplitude amplification to load coefficients calculated by an oracle, there has been a long series of results and improvements with various additional conditions on the amplitudes to be loaded, culminating in Sanders et al.'s work which avoids almost all arithmetic during the preparation stage.

In this work, we construct an optimized black box state loading scheme with which various important sets of coefficients can be loaded significantly faster than in $\mathrm{O}(\sqrt{N})$ rounds of amplitude amplification—up to only $\mathrm{O}(1)$ many. We achieve this with two variants of our algorithm. The first employs a modification of the oracle from Sanders et al., which requires fewer ancillas ($\log_2 g$ vs $g+2$ in the bit precision g), and fewer non-Clifford operations per amplitude amplification round within the context of our algorithm. The second utilizes the same oracle, but at slightly increased cost in terms of ancillas ($g + \log_2 g$) and non-Clifford operations per amplification round. As the number of amplitude amplification rounds enters as multiplicative factor, our black box state loading scheme yields an up to exponential speedup as compared to prior methods. This speedup translates beyond the black box case.

1 Introduction

The ability to prepare an arbitrary quantum state is a fundamental building block for many higher-level quantum algorithms, for instance in sparse linear algebra calculations [HHL09], quantum walks [Kem03; San08; BC09], machine learning and optimization [Bra+17; Bra+19], and quantum chemistry or condensed matter simulation [BCK15; Cod+12]. Starting with Grover [Gro00], a series of techniques have been developed, based on amplitude amplification [SS06; San+19; PB11] for the black box regime, or alternative approaches [Mot+04; Ara+20; GR02] in case the amplitudes are known à priori. Yet aims and limitations considered in these two regimes differ: generic black box algorithms allow loading amplitudes that stem from oracle subroutines, and others only prepare states for a particular set of amplitudes that has to be known from the outset. Furthermore, precision, requirements in terms of ancillas or intermediate measurements, or restrictions that have to be placed on the amplitudes to be loaded are not standardised, and vary from case to case.

In brief, quantum state preparation addresses the task to transform an amplitude vector $\alpha = (\alpha_0, \dots, \alpha_{N-1})$ into a quantum state close to

$$\frac{1}{\|\alpha\|_2} \sum_{i=0}^{N-1} \alpha_i |i\rangle \quad \text{or} \quad \frac{1}{\sqrt{\|\alpha\|_1}} \sum_{i=0}^{N-1} \sqrt{\alpha_i} |i\rangle .$$

The first "linear coefficient" problem, as dubbed by [San+19], is the one we consider in this work as the generic quantum state loading task. In this form, it is a crucial component e.g. for quantum linear algebra routines (HHL, to prepare an initial state $|b\rangle$ for which to solve $A|x\rangle = |b\rangle$), or random walks (to prepare an initial state whose amplitudes can be given by an arithmetic formula such as $\alpha_x \sim \sin(\pi(x+1)/(N+1))$ for a vertex x of a graph, [Chi09]).

If the amplitudes α_i are all known beforehand, circuits that hard-code the given data into gates can be developed [Mot+04; Ara+20]. As shown in [Mot+04], to load an N-amplitude state requires a circuit depth of $\Omega(N)$. In [Ara+20], the authors trade this depth (which is exponential in the number of qubits) for N ancillas, and an $\Theta(\log^2 N)$ depth; yet this tradeoff has to be taken with a grain of salt, as long-range gates, when broken down to an e.g. linear (or grid-like) qubit topology require $\sim N$ swaps to be performed throughout, and the resulting state is left entangled with the ancillas, i.e. a superposition of states $|i\rangle|\psi_i\rangle$ instead of just basis states $|i\rangle$. Furthermore, learning-based methods exist that yield circuits which approximately produce the target state, for instance based on generative adversarial neural networks [ZLW19]. Since the ansatz circuit can be chosen freely, depending on the required accuracy ciruit depths less than $\mathrm{O}(N)$ and without additional ancilla requirements can be achieved.

Instead of the case where the amplitude vector α is known from the outset, the focus in this paper is on black box algorithms based on oracles access to α: as is common, the procedure for creating such a state is based on the existence of an oracle $\mathbf{U}_\alpha$, which acts on a bipartite state

$$\mathbf{U}_\alpha |i\rangle |j\rangle = |i\rangle |j \oplus A_i\rangle , \tag{1}$$

where we assume A_i is a g-bit approximation to α_i, and $|j\rangle$ is a g-bit register. With the aid of amplitude amplification, this oracle is then raised to an overall procedure to produce a state with amplitudes close to α. To date, the best-known asymptotic runtime (measured in the number of necessary oracle calls) is $\sim \sqrt{N}$ [Gro00; SS06; San+19; PB11].

Our Contribution. In this paper, we improve upon the state-of-the art, i.e. Sanders et al.'s black box algorithm, by significantly improving loading times for black box quantum states if some prior knowledge about the amplitudes is known. This prior information is "cheaply" available from the oracle, in the form of the amplitudes' *average bit weight*: if the amplitudes are given as a sequence of binary numbers, the expected value of the j^{th} bit is simply the average over all j^{th} bits in the sequence. An approximation to these expected bit values can be measured quickly by a few oracle calls and one-qubit measurements. We prove that for various distributions this additional piece of information can significantly reduce the necessary number of amplitude amplification rounds, from $\mathrm{O}(\sqrt{N})$ to up to $\mathrm{O}(1)$, i.e. an up to *exponential speedup* for state preparation, and in various settings relevant for real-world scenarios (cf. table 2 and appendix A).

We develop our black box state loading algorithm based on a variant of the [San+19; GR02] oracle, which, within the context of our algorihtm, is significantly more frugal in terms of ancillas required for the same precision to be loaded: for instance for amplitudes with 32 bits of precision, [San+19] require at least 34 additional ancillas, whereas for our algorithm 5 suffice—at the cost of requiring a slightly deeper circuit.[1] As the fact that we utilize two different oracles means we are not comparing like to like, we also construct a variant of our algorithm based on *the same* oracle as in [San+19]. While this comes at a slight overhead in terms of ancillas and non-Clifford gates required (logarithmic in the precision of the amplitudes to be loaded, to be precise), the asymptotic speedups in terms of number of amplitude amplification rounds prevail and often outstrip the additional resource requirements necessary per round. This means that even in case

[1] We did not take any device topology into account in this comparison, where more ancillas often mean more reshuffling of information, with a correspondingly higher resulting circuit depth.

of utilizing the same oracle as [San+19] an exponential speedup is achievable, as compared to prior methods.

Notably, our proposed improvement goes *beyond* the black box setting. If the oracle is, for instance, an arithmetic circuit producing the weights $\alpha_i \sim \sin(\pi(i+1)/(N+1))$ for $i = 0, \ldots, N$ for a discrete quantum walk application, those weights might well be known beforehand, so pre-compiled state preparation circuits can be applied; but instead of a resulting depth $\Omega(N)$ circuit, already existing black-box algorithms reduce the asymptotics to $\mathrm{O}(\sqrt{N})$; our algorithm further reduces the runtime to $\mathrm{O}(1)$, translating the potential exponential speedup even to the case when amplitudes are known beforehand.

Finally, we generalise our state loading procedure as a generic subroutine able to prepare states like

$$\sum_{i=0}^{N-1} \left(\sum_{j=1}^{g} b_{ij} w_j \right) |i\rangle$$

for some boolean matrix $(b_{ij})_{1 \le i,j \le g}$ and weight vector $(w_j)_{1 \le j \le g}$, potentially useful as a linear algebra subroutine, and where the same optimisations for given prior knowledge about the b_{ij} mentioned in the last paragraph can be applied.

2 Gradient-State Based Black-Box Quantum State Preparation

The algorithm we propose shares similarities with [San+19]'s method, in the sense that it is a two-step protocol based on preparation of an intermediate state which maps the oracle's amplitudes into an ancillary system, and a second step that collates the ancillary amplitudes into the final form. In either step, amplitude amplification is employed to transform one state to the other.

Yet while Sanders et al. use a large ancillary register where the 2^g dimensions represent g bit numbers on a linear scale $0, 2^{-g}, 2 \times 2^{-g}, 3 \times 2^{-g}, \ldots, (2^g - 1) \times 2^{-g}$, we identify the ancillary dimensions with a logarithmic scale $1/2, 1/4, 1/8, \ldots, 2^{-g}$. This mapping has two advantages: i. it requires fewer qubits ($\lceil \log_2 g \rceil$ instead of g many), and ii. it allows knowledge of the oracle's average bit weight to be exploited, often reducing the necessary number of amplitude amplification rounds below Sanders et al. and Grover's $\mathrm{O}(\sqrt{N})$.

In section 2.1 we present the state loading protocol in detail, explaining the two stages, and deriving explicit runtime bounds. Section 2.2 then contains a discussion of error bounds and success probabilities, and in section 2.3 we describe how the new ingredients that our proposal is based on—preparing the logarithmic scale in the ancilla state, a reduction to the [San+19; GR02] oracle, and computing a swap network that replaces Sanders et al.'s comparator subroutine—can be constructed using few additional resources. Section 3 then discusses how some initial calls to the oracle can be used to produce an optimised initial state that results in a reduced number of necessary amplitude amplification rounds. We further present comparisons for number of ancillas and non-Clifford resources in table 1, and speedups obtained by using the optimised bootstrapping technique in table 2.

2.1 The Algorithm

Throughout the paper, $\|x\|_p := (\sum_i |x_i|^p)^{1/p}$ denotes the standard ℓ_p norm. Given a discrete N-element list of nonnegative amplitudes $\alpha = (\alpha_0, \ldots, \alpha_{N-1})$ with $\|\alpha\|_2 = 1$, we let $2^g A_i \in \mathbb{N}$ be a binary approximation to α_i to $g \in \mathbb{N}$ bits of precision (rounded towards zero), and let analogously $A = (A_0, A_1, \ldots, A_{N-1})$ be a g-bit approximation of the entire amplitude vector. The j^{th} bit of A_i is then denoted A_{ij}, in Little Endian order, i.e. such that A_{i0} is the most significant bit (denoting the 1/2's); with this we simply have $A_i = \sum_j 2^{-j} A_{ij}$. If not stated differently we assume that

$N = 2^n$ for some $n \in \mathbb{N}$; but the construction is generalised readily to non-power-of-2 numbers (e.g. by padding the vector α suitably by zeroes). Our goal is to prepare the state

$$|A\rangle = \frac{1}{\|A\|_2} \sum_{i=0}^{N-1} A_i |i\rangle . \tag{2}$$

How far does this state deviate from $|\alpha\rangle = \sum_{i=0}^{N-1} \alpha_i |i\rangle$, in trace distance $\sqrt{1 - |\langle \alpha|A\rangle|^2}$? We first note

$$|\langle \alpha|A\rangle| = \frac{1}{\|A\|_2} \sum_{i=0}^{N-1} \alpha_i A_i \overset{*}{\geq} \frac{1}{\|A\|_2} \sum_{i=0}^{N-1} A_i A_i = \|A\|_2 \geq \sqrt{1 - 2 \times 2^{-g} \|\alpha\|_1},$$

where in the step marked with $(*)$ we have made use of the fact that the A_i is rounded towards zero. Then

$$\sqrt{1 - |\langle \alpha|A\rangle|^2} \leq 2^{(1-g)/2} \sqrt{\|\alpha\|_1}. \tag{3}$$

In order to obtain a copy of this state $|A\rangle$ with high likelihood, we first define the amplitude gradient state

$$|g\rangle_G \coloneqq \frac{1}{\sqrt{2^g - 1}} \sum_{j=0}^{g-1} 2^{(g-j-1)/2} |j\rangle , \tag{4}$$

such that e.g.

$$|1\rangle_G = |0\rangle , \quad |2\rangle_G = \frac{1}{\sqrt{3}} \left(\sqrt{2} |0\rangle + |1\rangle \right) , \quad |3\rangle_G = \frac{1}{\sqrt{7}} \left(\sqrt{4} |0\rangle + \sqrt{2} |1\rangle + |2\rangle \right) , \quad \text{etc.}$$

We assume for now that this state can be prepared from a qudit ancilla $|0\rangle \in \mathbb{C}^g$ with a suitable unitary operation $\mathbf{G}$ (for implementation details see section 2.3.3), where we can again for simplicity assume that g is a power of 2. A combination of Hadamard operations and $\mathbf{G}$ then allows us to prepare the initial state

$$|s\rangle \coloneqq \left[\mathbf{H}^{\otimes n} \otimes \mathbf{G}\right] |0\rangle^{\otimes n} |0\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle |g\rangle_G . \tag{5}$$

The Grover oracle for this setup is given by a unitary operation, defined on basis states $|i\rangle |j\rangle \in (\mathbb{C}^2)^{\otimes n} \otimes \mathbb{C}^g$ via

$$\mathbf{U}_\omega |i\rangle |j\rangle = (-1)^{A_{ij}} |i\rangle |j\rangle . \tag{6}$$

In other words, $\mathbf{U}_\omega$ can compute bits of the coefficient α_i, and flips the sign indicating the j^{th} bit of the i^{th} index if and only if $A_{ij} = 1$.

For instance on the initial state $|s\rangle$, $\mathbf{U}_\omega$ takes the action

$$\mathbf{U}_\omega |s\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \times \frac{1}{\sqrt{2^g - 1}} \sum_{j=0}^{g-1} 2^{(g-j-1)/2} (-1)^{A_{ij}} |j\rangle .$$

There is various ways such an oracle unitary with phase kickback can be constructed; the most efficient means will depend on the target distribution α; for now we simply assume its existence. We emphasize that $\mathbf{U}_\omega$ is naturally not the same oracle as in [GR02; San+19] that can write out an entire amplitude $\mathbf{U}_{\text{amp}} |i\rangle |z\rangle = |i\rangle |z \oplus A_j\rangle$ in one query. While the black box state preparation algorithm we present in this paper is fundamentally based in $\mathbf{U}_\omega$, we discuss in section 2.3.1 how to proceed from $\mathbf{U}_{\text{amp}}$ as a starting point, which will incur both a qubit and gate overhead. In terms of query complexity to $\mathbf{U}_\omega$ or $\mathbf{U}_{\text{amp}}$, however, the two settings will prove to be identical.

The final ingredient is a variant of the Grover diffusion operator, namely

$$\mathbf{U}_s \coloneqq \left[\mathbf{H}^{\otimes n} \otimes \mathbf{G}\right] \left[2 |0\rangle\langle 0| - \mathbb{1}\right] \left[\mathbf{H}^{\otimes n} \otimes \mathbf{G}\right]^\dagger . \tag{7}$$

Defining the intermediate target state

$$|\omega\rangle \coloneqq \frac{1}{\sqrt{\|A\|_1}} \sum_{i=0}^{N-1} |i\rangle \sum_{j=0}^{g-1} 2^{-(j+1)/2} A_{ij} |j\rangle \tag{8}$$

we can—as a well-known step—equate both $\mathbf{U}_s$ and $\mathbf{U}_\omega$ with two Householder operations.

Lemma 1. $\mathbf{U}_s = 2|s\rangle\langle s| - \mathbb{1}$, *and* $\mathbf{U}_\omega = \mathbb{1} - 2|\omega\rangle\langle\omega|$ *when restricted to the subspace spanned by* $|s\rangle$ *and* $|\omega\rangle$.

Proof. By eq. (7), we have that

$$\mathbf{U}_s = 2\left(\left[\mathbf{H}^{\otimes n} \otimes \mathbf{G}\right]|0\rangle\langle 0|\left[\mathbf{H}^{\otimes n} \otimes \mathbf{G}\right]^\dagger\right) - \mathbb{1} = 2|s\rangle\langle s| - \mathbb{1}$$

by definition of the initial state $|s\rangle$ in eq. (5). To prove that $\mathbf{U}_\omega$ is also a Householder transformation within the subspace spanned by the initial state $|s\rangle$ (eq. (5)) and the intermediate target state $|\omega\rangle$ (eq. (8)), we proceed by direct calculation.

$$\begin{aligned}
\mathbf{U}_\omega |s\rangle &= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \times \frac{1}{\sqrt{2^g - 1}} \sum_{j=0}^{g-1} 2^{(g-j-1)/2} (-1)^{A_{ij}} |j\rangle \\
&\overset{*}{=} |s\rangle + \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \times \frac{1}{\sqrt{2^g - 1}} \sum_{j=0}^{g-1} \left[2^{(g-j-1)/2} (-1)^{A_{ij}} |j\rangle - 2^{(g-j-1)/2} |j\rangle\right] \\
&= |s\rangle - \frac{2}{\sqrt{N}} \frac{1}{\sqrt{2^g - 1}} \sum_{i=0}^{N-1} |i\rangle \sum_{j=0}^{g-1} 2^{(g-j-1)/2} A_{ij} |j\rangle \\
&= |s\rangle - 2\frac{\sqrt{\|A\|_1}}{\sqrt{N}} \frac{2^{g/2}}{\sqrt{2^g - 1}} |\omega\rangle ,
\end{aligned} \tag{9}$$

where in the step marked with $*$ we have added and subtracted $|s\rangle$; the term on the right hand side of the square brackets in that line stems from the summand in the gradient state, eq. (4). As

$$\begin{aligned}
\langle\omega|s\rangle &= \frac{1}{\sqrt{N}\sqrt{\|A\|_1}} \sum_{i=0}^{N-1} \sum_{i'=0}^{N-1} \langle i|i'\rangle \frac{1}{\sqrt{2^g - 1}} \sum_{j=0}^{g-1} \sum_{j'=0}^{g-1} 2^{(g-j'-1)/2} 2^{-(j+1)/2} A_{ij} \langle j|j'\rangle \\
&= \frac{1}{\sqrt{N}\sqrt{\|A\|_1}} \frac{2^{g/2}}{\sqrt{2^g - 1}} \sum_{i=0}^{N-1} \underbrace{\sum_{j=0}^{g-1} 2^{-j-1} A_{ij}}_{\equiv A_i} \\
&= \frac{\sqrt{\|A\|_1}}{\sqrt{N}} \frac{2^{g/2}}{\sqrt{2^g - 1}},
\end{aligned} \tag{10}$$

we see that eq. (9) further simplifies to $\mathbf{U}_\omega |s\rangle = |s\rangle - 2\langle\omega|s\rangle |w\rangle$.

The last operation to verify is

$$\mathbf{U}_\omega |\omega\rangle = \frac{1}{\sqrt{\|A\|_1}} \sum_{i=0}^{N-1} |i\rangle \sum_{j=0}^{g-1} 2^{-(j+1)/2} A_{ij} (-1)^{A_{ij}} |j\rangle = -|\omega\rangle ,$$

as $A_{ij}(-1)^{A_{ij}} = -A_{ij}$. □

In combination, lemma 1 thus forms the Grover iteration operator $\mathbf{U}_s\mathbf{U}_\omega$, which performs a rotation in the space spanned by $|s\rangle$ and $|\omega\rangle$.

We observe that the fact that lemma 1 only shows that $\mathbf{U}_\omega$ has Householder form only when restricted to this hyperplane is also the case for normal Grover search, e.g. in the case when there are multiple marked target elements. Apart from the explicit proof we just gave for lemma 1, it

is easy to see that this must be the case, as the oracle $\mathbf{U}_\omega$ given in eq. (6) has precisely the same form as the standard Grover oracle; the explicit form of the two vectors $|s\rangle$ and $|\omega\rangle$ thus emerge from the Grover diffusion operator $\mathbf{U}_s$, which implicitly defines the initial state $|s\rangle$.

Starting from this initial state $|s\rangle$, we can thus use these Grover iterations to approach the intermediate target state $|\omega\rangle$ step-by-step. Instead of deriving the runtime of this variant of Grover search (which is by now standard and can e.g. be found in [NC10, Sec.6.1.3]), we rely on the state-of-the-art in fixed-point amplitude amplification in order to not overshoot the target [YLC14]. We cite their result here in a concise form for completeness.

Lemma 2 (Fixed-Point Amplitude Amplification, [YLC14])**.** *Let $n \in \mathbb{N}$. We are given a unitary operator $\mathbf{S}$ that prepares an initial state $|s\rangle = \mathbf{S}\,|0\rangle^{\otimes n}$, and a target state $|T\rangle$ with $\langle T|s\rangle = \sqrt{\lambda}\mathrm{e}^{\xi}$, $\lambda, \xi \in \mathbb{R}$, and an oracle $\mathbf{U}\,|T\rangle\,|b\rangle = |T\rangle\,|b \oplus 1\rangle$ and $\mathbf{U}\,|\bar{T}\rangle\,|b\rangle = |\bar{T}\rangle\,|b\rangle$ for $\langle \bar{T}|T\rangle = 0$. Then there exists a fixed point amplitude amplification procedure that, when starting from the initial state $|s\rangle$, extracts the target state $|T\rangle$ with success probability $\geq 1 - \delta^2$ for some $\delta \in [0, 1]$, utilizing $L \sim \log(2/\delta)/\sqrt{\lambda}$ calls to the oracle $\mathbf{U}$ interleaved by $2L$ calls to $\mathbf{S}$.*

A success probability of $1 - \delta^2$ means that after the given number of rounds the fixed-point amplification procedure results in a state $|T'\rangle$ which satisfies

$$1 - \delta^2 = \mathrm{Tr}(|T\rangle\langle T|\,|T'\rangle\langle T'|) = |\,\langle T|T'\rangle\,|^2.$$

For us, $\mathbf{S} = \mathbf{H}^{\otimes n} \otimes \mathbf{G}$, as is clear from eqs. (5) and (7), and the oracle $\mathbf{U} = \mathbf{U}_\omega$ is the same with a phase kickback mechanism implied (as already explained). Further translating this terminology to our setting, we start with the initial state $|s\rangle$, and amplify to an intermediate target state $|T\rangle = |\omega\rangle$; where then $|\omega'\rangle$ denotes the state we *actually* reach after the amplitude amplification has been applied.

Denoting the number of fixed point amplification rounds with L_1 (where the subscript indicates that our overall algorithm will have a secondary stage with similar quantities subscripted by 2, to be determined in due course), we need

$$L_1 \sim \frac{\log(2/\delta_1)}{\sqrt{\lambda_1}} \quad \text{where} \quad \delta_1 = \sqrt{1 - |\,\langle\omega|\omega'\rangle\,|^2} \quad \text{and} \quad \lambda_1 = |\,\langle\omega|s\rangle\,|^2. \tag{11}$$

The parameter δ_1 is thus a guaranteed trace distance to the intermediate target state that we will reach; and λ_1 indicates the overlap of the target coefficient vector α with the initial state $|s\rangle$.

Then with eqs. (3) and (10) we have

$$\sqrt{\lambda_1} = \langle\omega|s\rangle = \sqrt{\frac{\|A\|_1}{N}}\sqrt{\frac{2^g}{2^g - 1}} \geq \sqrt{\frac{\|\alpha\|_1 - 2^{-g}N}{N}}. \tag{12}$$

The two extremes here are the cases where α itself is already uniform—such that $\alpha_i = 1/\sqrt{N}$, which yields $L_1 = \mathrm{O}(N^{1/4})$; and a delta distribution—which yields $L_1 = \mathrm{O}(N^{1/2})$. The reason why the uniform example does not already give $L_1 = \mathrm{O}(1)$ is that $|s\rangle$ does *not* represent a uniform intermediate target state—in fact, $|s\rangle$ just is a uniform superposition of all possible bits, whereas a uniform superposition will likely have $A_{ij} = 0$ for a series of higher-significance positions. We improve upon this caveat in section 3.

How do we move from the intermediate target state $|\omega\rangle$ from eq. (8) to the target state $|A\rangle$? We must project the second register onto the gradient state; either by postselection, or by another nested round of amplitude amplification. We note that the fidelity of amplification in this step only increments the probability of obtaining the state $|A\rangle$; if we succeed, we are guaranteed that we are left with that exact state, and there is no more introduced error.

If we perform amplitude amplification, we denote with L_2, δ_2 and λ_2 the associated amplitude amplification parameters from eq. (11). If we define the projector $\Pi \coloneqq \mathbb{1} \otimes |g\rangle\langle g|_G$, we can calculate

$$\begin{aligned}\lambda_2 = \langle\omega|\,\Pi\,|\omega\rangle &= \frac{1}{\|A\|_1}\frac{1}{2^g-1}\sum_{i=0}^{N-1}\left(\sum_{j=1}^{g} 2^{(g-j)/2}2^{-j/2}A_{ij}\right)^2 \\ &= \frac{1}{\|A\|_1}\frac{2^g}{2^g-1}\sum_{i=0}^{N-1}(A_i)^2 \geq \frac{\|A\|_2^2}{\|A\|_1}. \end{aligned} \tag{13}$$

As $A_i \leq \alpha_i$ (since we round towards zero), we have that

$$\frac{\|A\|_2^2}{\|A\|_1} \geq \frac{\|\alpha - 2^{-g}\|_2^2}{\|\alpha\|_1} = \frac{1}{\|\alpha\|_1} - 2\times 2^{-g} + \frac{2^{-2g}N}{\|\alpha\|_1} \geq \frac{1}{\|\alpha\|_1} - 2\times 2^{-g}.$$

With $\langle\omega|\,\Pi\,|\omega\rangle \approx 1/\|\alpha\|_1$, as before, we observe that the 1-norm of the coefficients α determines the overlap, with extreme cases being the uniform distribution for which $\|\alpha\|_1 = \sqrt{N}$, and delta distribution for which $\|\alpha\|_1 = 1$.

Jointly together these two overlaps yield the overall runtime: we have

$$\begin{aligned}\lambda_1\lambda_2 &\geq \frac{\|\alpha\|_1 - 2^{-g}N}{N}\left(\frac{1}{\|\alpha\|_1} - 2\times 2^{-g}\right) = \frac{1}{N} - 2^{-g}\left(\frac{1}{\|\alpha\|_1} - 2\frac{\|\alpha\|_1}{N} - 2\times 2^{-g}\right) \\ &\geq \frac{1}{N} - \frac{2^{-g}}{\|\alpha\|_1}.\end{aligned}$$

As the two amplitude amplification subroutines have to be called in a nested fashion, we have

$$\begin{aligned}L = L_1L_2 = \frac{\log(2/\delta_1)\log(2/\delta_2)}{\sqrt{\lambda_1\lambda_2}} &\leq \log(2/\delta_1)\log(2/\delta_2)\Big/\sqrt{\frac{1}{N} - \frac{2^{-g}}{\|\alpha\|_1}} \\ &\overset{*}{\leq} \log(2/\delta_1)\log(2/\delta_2)\left(\sqrt{N} + \frac{2^{-g}N}{\|\alpha\|_1}\right),\end{aligned}$$

where the last inequality marked with $(*)$ holds for high enough precision g such that $x = 2^{-g}N/\|\alpha\|_1 < 1/2$, and using $(1-x)^{-1} \leq 1 + x$ for $x \leq 1/2$.[2] Under this assumption, we have

$$L = L_1L_2 \leq \log(2/\delta_1)\log(2/\delta_2)\left(\sqrt{N} + \frac{1}{2}\right). \tag{14}$$

2.2 Precision and Error Analysis

For runtime estimates we already kept track of lower bounds as given in eq. (14). Furthermore, we can independently tune the success probability δ_2 which determines with what likelihood we obtain a copy of the state $|A\rangle$ (or, more precisely, the state $|A'\rangle$ that we obtain when starting from an approximate intermediate target state $|\omega'\rangle$ instead of $|\omega\rangle$ in the previous amplification round), and depending on what failure rate we find acceptable; in other words, δ_2 determines with what likelihood we obtain a state $|\psi\rangle$, δ_1 controls how close $|\psi\rangle$ to $|A\rangle$ a state we obtain, and the parameter g controls the precision of the coefficients (i.e. how close $|A\rangle$ is to $|\alpha\rangle$).

As our overall procedure approximates $|\alpha\rangle$ by $|A\rangle$, which by eq. (3) differ in trace distance by $\leq 2^{(1-g)/2}\sqrt{\|\alpha\|_1}$, we should choose δ_1 to represent a trace distance of about the same amount (resulting in $\sqrt{2}$ times this distance overall). Then eq. (14) reads

$$L \leq \log\left(\frac{2}{\delta_2}\right) \times \frac{1}{2}\big(1 + g - \log\|\alpha\|_1\big) \times \left(\sqrt{N} + \frac{1}{2}\right).$$

[2] We note that this bound can be made tighter by a constant factor up to two, by expanding $(1-x)^{-1} \leq 1 + yx$ for some $y \in (1/2, 1]$ and correspondingly demanding $x < (y + \sqrt{y(4+y)} - 2)/(2y)$.

	bit precision g	2	4	8	16	32 (>float)	64 (>double)
	$\epsilon = 2^{-g} \leq$	$\frac{1}{4}$	$\frac{1}{16}$	0.004	1.6×10^{-5}	2.3×10^{-10}	5.5×10^{-20}
Toffolis	[San+19] variant 1	4	8	16	32	64	128
	[San+19] variant 2	6	14	30	62	126	254
	us, variant 1	none					
	us, variant 2	3	12	33	78	171	360
$\sqrt{\text{SWAP}}$	[San+19]	none					
	us	2	4	8	16	32	64
ancillas	[San+19] variant 1	5	9	17	33	65	129
	[San+19] variant 2	4	6	10	18	34	66
	us, variant 1	1	2	3	4	5	6
	us, variant 2	3	6	9	20	37	70

Table 1: Reachable precision for a given gradient state $|g\rangle_G$ on q qubits, and non-Clifford gate counts for each amplitude amplification iteration. [San+19]'s variant 1 and 2 refer to the two options for implementing their comparator circuits ($2g+1$ ancillas, $2g$ Toffolis; resp. $g+2$ ancillas, $4g-2$ Toffolis). Our two variants refer to utilizing $\mathbf{U}_\omega$ or $\mathbf{U}_{\text{amp}}$ as oracle, as discussed in section 2.3.2 ($\log_2 g$ ancillas, no Toffolis; resp. $g + \log_2 g$ ancillas, $\leq 2g \log_2 g$ Toffolis; and g $\sqrt{\text{SWAP}}$s in either case for the gradient state). Our SWAP network for variant 2 is built as in fig. 1, and trivial gate optimizations are applied. As [San+19] use $\mathbf{U}_{\text{amp}}$, only our variant 2 should be directly compared to [San+19]. We refer the reader to section 3 and table 2 for a speedup with regards to amplitude amplification rounds necessary. Device topology is not taken into account, where for planar architectures it is often the case that more ancillas mean more reshuffling of information, with correspondingly higher circuit depth.

This is in alignment with [Gro00; San+19], who quote an approximate runtime bound for their "linear coefficients" problem of $O(g\sqrt{N})$, where the authors omitted the logarithmic error terms due to δ_1 and δ_2. One can verify that keeping track of amplitude amplification errors throughout their procedure yields equivalent factors.

As the gradient state can store g values within $\lceil \log_2 g \rceil$ many qubits, we can load high-precision numbers with little overhead, as shown in table 1.

2.3 Computational Primitives and Resource Requirements

There are two fundamental building blocks that we need to analyse: implementing an amplitude gradient gate for states like eq. (4), and the phase kickback oracle unitary $\mathbf{U}_\omega$ from eq. (6) in case we only have access to an oracle as in [San+19; GR02].

2.3.1 Oracle Comparison between $\mathbf{U}_\omega$ and $\mathbf{U}_{\text{amp}}$

Grover and Rudolph; Sanders et al. analyse the complexity of their black box state preparation algorithms by assuming there exists an oracle unitary $\mathbf{U}_{\text{amp}}$ that can calculate the desired amplitudes digitally, as

$$\mathbf{U}_{\text{amp}} |i\rangle |z\rangle = |i\rangle |z \oplus A_i\rangle \tag{15}$$

for any bit string $z \in \{0,1\}^g$ and computational basis state $|i\rangle$. How this oracle unitary is implemented is left unspecified; in a sense this notion completely decouples the aspect of *computing* the coefficients from *loading* these coefficients as actual amplitudes. We will keep with this notion, as

the oracle's own resource requirements are highly dependent on the amplitudes to be loaded. For similar reasons, a direct comparison of the resource requirements between $\mathbf{U}_\omega$ and $\mathbf{U}_{\text{amp}}$ is futile.

In addition, comparing two black box state loading algorithms that depend on different oracles is only fair if neither of the oracles yields an unfair advantage over the other. To rule this out, we will show in the following discussion that we can phrase a variant of our black box state loading procedure that uses $\mathbf{U}_{\text{amp}}$ as an oracle instead of $\mathbf{U}_\omega$, which will put both algorithms on a level playing field. This will induce a minor overhead in terms of ancillas, but crucially results in the same query complexity, as exactly one application of $\mathbf{U}_{\text{amp}}$ will be required to replace one application of $\mathbf{U}_\omega$.

In conjunction with the oracle, Sanders et al. employ a non-Clifford step that requires a comparator circuit mediated by a conditional flip of a flag qubit

$$|A\rangle\,|B\rangle\,|0\rangle \longmapsto |A\rangle\,|B\rangle \begin{cases} |1\rangle & A \geq B \\ |0\rangle & A < B \end{cases} \tag{16}$$

which is used to transduce the amplitudes into an ancillary system (cf. [San+19, eq. 6]).

In contrast, the algorithm we introduced in this paper assumes the existence of an oracle $\mathbf{U}_\omega$ (eq. (6)), which induces a conditional phase flip $\mathbf{U}_\omega\,|i\rangle\,|j\rangle = (-1)^{A_{ij}}\,|i\rangle\,|j\rangle$. When applied to a gradient state in place of register $|j\rangle$, one application of $\mathbf{U}_\omega$ thus achieves that a total weight of coefficients proportional to A_i has a flipped coefficient, which is equivalent to the combined step in Sanders et al. Again, it is important to emphasize that this does not mean the oracles are equivalent, or that query complexity for the two routines should be comparable a priori.

However, in section 2.3.2 we will demonstrate directly that with the aid of a comparator circuit one can utilize $\mathbf{U}_{\text{amp}}$ to implement $\mathbf{U}_\omega$; and thus obtain a variant of our proposed algorithm based on $\mathbf{U}_{\text{amp}}$ as fundamental oracle primitive. As this renders the algorithms more comparable, we provide a resource calculation for either choice of oracle in table 2.

2.3.2 Comparator Circuits, and Replacing $\mathbf{U}_\omega$ with $\mathbf{U}_{\text{amp}}$

In Sanders et al.'s comparator circuit, comparing two g bit numbers as in eq. (16) can be done using g Toffoli gates [Cuc+04; Gid18]. As the method for this comparison relies on binary addition (or rather subtraction) which in this case is performed in-place, the register $|A_i\rangle$ first has to be copied to a temporary slot (which is always possible as it is a computational basis state). The comparator itself thus requires $g+1$ ancillas, and $2g$ Toffoli gates (as the comparison subroutine has to be uncomputed as well).

In our case, if we amend our algorithm to be based on $\mathbf{U}_{\text{amp}}$ instead of $\mathbf{U}_\omega$, we first apply

$$\mathbf{U}_{\text{amp}}\frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle\,|0\rangle\,|g\rangle_G = \frac{1}{\sqrt{N}}\frac{1}{\sqrt{2^g-1}}\sum_{i=0}^{N-1}|i\rangle\,|A_i\rangle\sum_{j=0}^{g-1}2^{(g-j-1)/2}\,|j\rangle\,.$$

As the register $|j\rangle$ stems not from uniformly-distributed bins, but from a binary tree partition (i.e. from the gradient state $|g\rangle_G$) the next step is to introduce a phase flip on $|j\rangle$ if the j^{th} bit of A_i is 1, i.e. when $A_{ij}=1$.

As aforementioned, with the use of g ancillas it is straightforward to translate $|j\rangle$ into a unary one-hot mask and then perform such a bit-by-bit comparison with $2g$ Toffoli gates, which matches the bound given in [San+19].

If we want to be qubit-conservative (and assuming $|j\rangle$ remains stored in binary), one can alternatively decompose a $\lceil\log_2 g\rceil+1$-control Toffoli gate into primitive resources (to e.g. check the address register $|j\rangle=|5\rangle$, and that $A_{i5}=1$, and conditionally flip a single ancilla). This can be done with $32\lceil\log_2 g\rceil-96$ $\mathbf{T}$ or $\mathbf{T}^\dagger$ gates [He+17]. This method results in roughly $8\log_2 g$ times the Toffoli gate count as in [San+19].

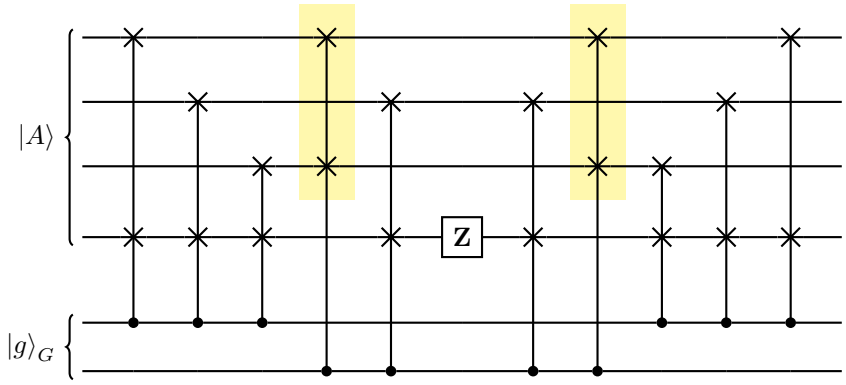

Figure 1: Permutation network for shuffling the x^{th} qubit of the register $|A\rangle$ to its least significant position, applying a phase flip there, and shuffling the qubits back to their original position. Shown is the example $g = 2$, which yields 4 bits of precision for the data to be loaded, as shown in table 1. The network is straightforward to generalize for larger g, and requires at most $2g\lceil \log_2 g \rceil$ controlled SWAP operations; in many cases, further optimizations can be applied (in this case the two shaded swaps are redundant because they are outside of the causality cone of the **Z**-gate, cf. table 1). A controlled SWAP operation—or Fredkin gate—can be implemented using a single Toffoli and two CNOTs [SD96].

A better approach is to use a custom permutation network as shown in fig. 1, which shuffles the j^{th} bit to the bottom position, where a **Z** operation can be performed (in essence using the register where $|A\rangle$ is written as a source for phase kickback). This results in the use of $2g \log_2 g$ Toffoli gates, i.e. an overhead of $\log_2 g$ as compared to [San+19]; the resulting SWAP networks can often be optimized further, by considering the light cone of the phase flip **Z** that is applied to the j^{th} bit. We summarise our resulting Toffoli counts in table 1.

As a final observation we note that if the oracle $\mathbf{U}_\omega$ is implemented such that one can query individual bits of the vector A to be loaded, no comparison is necessary at all; while this looks like sweeping unwanted complexity of the algorithm under the rug (it is certainly reasonable to assume that in order to calculate the k^{th} binary digit of a number one has to also compute all previous $k-1$ bits), it is educational to discuss this variant of the query model, as it demonstrates that the choice of oracle (even if they serve a similar purpose) dictates the query cost of an algorithm. In the same light, if we were to utilize $\mathbf{U}_\omega$ as a primitive in [San+19]'s work, we would have to query $\mathbf{U}_\omega$ at least g times in order to obtain a single amplitude written out in binary, what $\mathbf{U}_{\text{amp}}$ is capable of achieving in one step.

2.3.3 Amplitude Gradient State

Let us finally turn our attention to the gate complexity of preparing an amplitude gradient state $|g\rangle_G$ as in eq. (4). Assuming for now that the state space is that of a qudit $\mathbb{C}^g$, and with access to arbitrary rotations around **X**—i.e. gates of the form $\mathbf{U}_{ij}(\theta) = \exp(\mathrm{i}\theta\mathbf{X}_{ij})$ for $\theta \in [0, 2\pi)$ and where $\mathbf{X}_{ij}$ generates rotations in the subspace spanned by $|i\rangle$ and $|j\rangle$, we can iteratively apply

$$\begin{aligned} |0\rangle &\xmapsto{\mathbf{U}_{01}(\theta_1)} \cos\theta_1 |0\rangle + \sin\theta_1 |1\rangle \\ &\xmapsto{\mathbf{U}_{12}(\theta_2)} \cos\theta_1 |0\rangle + \sin\theta_1 (\cos\theta_2 |1\rangle + \sin\theta_2 |2\rangle) \\ &\;\;\vdots \end{aligned}$$

with a suitable sequence of angles θ_i. Yet implementing arbitrary single qubit rotations again requires non-Clifford gates, and compiling qudit operations down to the $\lceil \log_2 g \rceil$-sized qudit register that $|g\rangle_G$ resides on likely requires even more such resources.

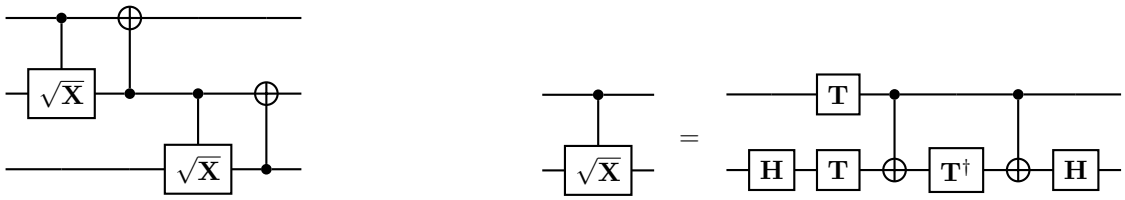

Figure 2: Left: $\sqrt{\text{SWAP}}$ chain used to create a $|g\rangle_G$ state, as described in section 2.3.3. Right: implementation of $\sqrt{\text{CNOT}}$ using 3 $\mathbf{T}$-gates.

An easier way of creating an amplitude gradient state for $g = 2^i - 1$ is if we allow ourselves to sacrifice a single dimension as slack space, create a state

$$\frac{1}{2^g}\left(2^{(g-1)/2}\,|0\rangle + 2^{(g-2)/2}\,|1\rangle + \ldots + \sqrt{2}\,|g-1\rangle + |g\rangle + |g+1\rangle\right) \tag{17}$$

and ignore the slack dimension $|g+1\rangle$ in all subsequent operations; to see that this does not introduce an additional error, note that the entire procedure simply acts as if the amplitudes A_i are really given to $g+1$ bits of precision—where the last bit has twice the weight as the second-to-last—but it just so happens that all $A_{i,g+1} = 0$.

A state like eq. (17) is readily prepared if we have access to a $g+1$-sized ancilla register; note that we do indeed have access to a g-sized space already, since whenever we apply the gate $\mathbf{G}$, the register storing the amplitudes A_i is reset to $|0\rangle^{\otimes g}$. In this case, with the aid of successive $\sqrt{\text{CNOT}_{12}}$-gates and CNOT_{21}, and e.g. for $g = 4$, we perform the operation

$$\begin{aligned}
|1000\rangle &\longmapsto \frac{1}{\sqrt{8}}\left(\sqrt{4}\,|1000\rangle + \sqrt{4}\,|0100\rangle\right) \\
&\longmapsto \frac{1}{\sqrt{8}}\left(\sqrt{4}\,|1000\rangle + \sqrt{2}\,|0100\rangle + \sqrt{2}\,|0010\rangle\right) \\
&\longmapsto \frac{1}{\sqrt{8}}\left(\sqrt{4}\,|1000\rangle + \sqrt{2}\,|0100\rangle + |0010\rangle + |0001\rangle\right).
\end{aligned}$$

We then convert the unary representation into a binary representation in the $|g\rangle_G$-state, e.g. $|0100\rangle\,|00\rangle_G \mapsto |0100\rangle\,|10\rangle_G$, which is straightforward with a sequence of hard-coded CNOT gates. To finally unentangle the ancilla register, we apply half of the permutation network in fig. 1 which results in a state $|0001\rangle\,|g\rangle_G$, and reset the remaining $|1\rangle$ to $|0\rangle$.

The Toffoli cost of this amplitude gradient state preparation protocol is thus equal to half the Toffoli cost of a $g+1$-bit permutation network; and g $\sqrt{\text{CNOT}_{12}} = \mathbf{H}_2\sqrt{\mathbf{cZ}_{12}}\mathbf{H}_2 = \mathbf{H}_2\mathbf{cS}_{12}\mathbf{H}_2$, where $\mathbf{cS}$ is a controlled-$\mathbf{S}$ gate that can be implemented as $\mathbf{cS}_{12} = (\mathbf{T}_1 \otimes \mathbf{T}_2)\text{CNOT}_{12}\mathbf{T}_2^\dagger\text{CNOT}_{12}$. The procedure is shown in fig. 2.

We also note that existing near-term quantum devices sometimes feature a variant of SWAP^θ gates natively, e.g. through the fSim gate [Goo+19, Eq. 53], and the above sequence of successively moving the single $|1\rangle$-ancilla to the right with a $1/\sqrt{2}$ factor can simply be implemented directly using g $\sqrt{\text{SWAP}}$ operations.

3 Optimized Bootstrapping

The amplitude amplification overhead from eq. (11) and the runtime bound in eq. (14) indicate that even if our aim was to load a uniform distribution, there would be a $L_1 L_2 \approx \sqrt{N}$ cost; this is not an artefact of our construction, but an overhead already present in Grover's original black-box

state loading paper. But we already start with a "uniform" initial state, $|s\rangle$—so where does this overhead originate?

The point to look at is the gradient state $|g\rangle$ from eq. (4), which doesn't serve as a good initial state for all possible amplitude vectors to be loaded. In fact, it weighs every bit as equally likely to occur in the final state to be loaded. What we should choose instead is a gradient state representing the most likely bit configuration to occur, weighted by the significance of the bit. For instance if $N = 64$ and $\alpha_i = 1/8$ for all i, then $A_i = \texttt{0.001}$, and the best initial vector thus consists of a single "on" bit, namely the $1/8^{\text{th}}$ position $|\alpha, 0\rangle = 0\,|0\rangle + 0\,|1\rangle + |2\rangle$. In this example no amplitude amplification round is necessary at all, since $|s\rangle$ is already a uniform superposition.

In brief, what we need to choose is an initial state $|\beta\rangle = \sum_{j=0}^{g-1} \beta_j\,|j\rangle$ such that the overlap with the intermediate target state is maximized, i.e.

$$\sum_{i=0}^{N-1} \langle\beta| \sum_{j=0}^{g-1} 2^{-j/2} A_{ij}\,|j\rangle = \sum_{j=1}^{g} \beta_j 2^{-j/2} \sum_{i=0}^{N-1} A_{ij} =: \sum_{j=1}^{g} \beta_j \bar{A}_j.$$

This is the case if $|\beta\rangle$ is chosen parallel to the vector defined by the coefficients $\bar{A}_j$ labelling the average j^{th} bit weight. We thus set

$$\left|\bar{A}\right\rangle \coloneqq \frac{1}{\|\bar{A}\|_2} \sum_{j=0}^{g-1} \bar{A}_j\,|j\rangle\,. \tag{18}$$

As an example, consider the case where we aim to load a quantum state with eight coefficients to three bits of precisison, and these coefficients are

$$\texttt{0.100},\quad \texttt{0.100},\quad \texttt{0.100},\quad \texttt{0.100},\quad \texttt{0.111},\quad \texttt{0.101},\quad \texttt{0.110},\quad \texttt{0.110}.$$

Then the first bit position is "on" in 8/8 cases, the second in 3/8 cases, and the third in 2/8 cases; thus $\bar{A}_0 = 1, \bar{A}_1 = 0.375$, and $\bar{A}_2 = 0.25$. The normalization in eq. (18) then renders this a valid quantum state.

With this new initial state $|s'\rangle = N^{-1/2} \sum_{i=0}^{N-1} |i\rangle \left|\bar{A}\right\rangle$ instead of eq. (5), we obtain a shorter L_1' time to amplify towards the intermediate target state $|\omega\rangle$, namely due to

$$\begin{aligned}
\sqrt{\lambda_1'} = \langle\omega|s'\rangle &= \frac{1}{\sqrt{N}} \frac{1}{\sqrt{\|A\|_1}} \frac{1}{\|\bar{A}\|_2} \sum_{i=0}^{N-1} \sum_{j=0}^{g-1} 2^{-(j+1)/2} A_{ij} \bar{A}_j \\
&= \frac{1}{\sqrt{N}} \frac{1}{\sqrt{\|A\|_1}} \frac{1}{\|\bar{A}\|_2} \underbrace{\sum_{j=0}^{g-1} \bar{A}_j \bar{A}_j}_{\equiv \|\bar{A}\|_2^2} \\
&= \frac{1}{\sqrt{N}} \frac{\|\bar{A}\|_2}{\sqrt{\|A\|_1}}.
\end{aligned} \tag{19}$$

In conjunction with the unaltered second state loading stage described in section 2.1 and eq. (13), we have

$$\begin{aligned}
\lambda_1' \lambda_2 &\geq \frac{1}{N} \frac{\|\bar{A}\|_2^2}{\|A\|_1} \frac{\|A\|_2^2}{\|A\|_1} \geq \frac{\|\bar{A}\|_2^2}{N} \frac{\|\alpha - 2^{-g}\|_2^2}{\|\alpha\|_1^2} \\
&\geq \frac{\|\bar{A}\|_2^2}{N} \frac{1 - 2 \times 2^{-g} \|\alpha\|_1}{\|\alpha\|_1^2} = \frac{1 - 2^{1-g}\|\alpha\|_1}{N} \left(\frac{\|\bar{A}\|_2}{\|\alpha\|_1}\right)^2.
\end{aligned}$$

We thus obtain an optimized state loading protocol with runtime

$$L' = L_1' L_2 \leq \log(2/\delta_1) \log(2/\delta_2) (1 + 2^{1-g} \|\alpha\|_1) \times \sqrt{N} \frac{\|\alpha\|_1}{\|\bar{A}\|_2}. \tag{20}$$

distribution	$L' = \mathrm{O}(\cdot)$	L for $N = \ldots$ 10^2	10^4	10^6	L' for $N = \ldots$ 10^2	10^4	10^6	
delta	$\sqrt{N}$	10	100	10^3	10	100	10^3	
uniform	1	⋮	⋮	⋮	1	1	1	
triangle $\alpha_i \propto i$	1	⋮	⋮	⋮	1	1	1	
powerlaw† $\alpha_r \propto r^{-k}$ (Fig. 3)	$N^{\delta(k)}$ with $\delta(k) < 1/2$	⋮	⋮	⋮	2	4	9	$k = 1/2$
					3	9	27	$k = 3/2$
					4	25	205	$k = 2$
normal† $\mathcal{N}(0, \sigma)$ (Fig. 4)	$\begin{cases} 1 & N \lessapprox \sigma \\ \sqrt{N/\sigma} & \text{otherwise} \end{cases}$	⋮	⋮	⋮	1	1	1	$\sigma = 10^4$
					1	3	22	$\sigma = 100$
					5	44	440	$\sigma = 1$
random† $\alpha_i \sim \mathrm{unif}(0, 1)$ (Fig. 5)	1	⋮	⋮	⋮	1	1	1	

Table 2: Speedups due to optimized initial state for state loading, as compared to the generic $\sqrt{N}$ runtime. Explicit calculations are given in appendix A; for those marked with †, runtimes are conjectured based on empirical analysis and unlikely to be tight.

For a discretized set of amplidutes like A, eq. (19) gives the exact expression for $\sqrt{\lambda_1'}$. Nevertheless, the quantity $\|\bar{A}\|_2$—i.e. the 2-norm of the root of the average bit weight—appears hard to get a handle on, in general. Yet we emphasise that in the black-box state loading picture where we already assume the existence of the oracle unitary $\mathbf{U}_{\mathrm{amp}}$ as in eq. (15) anyways (or $\mathbf{U}_\omega$ as in eq. (6)), with the aid of which estimating $\bar{A}_j$—i.e. the average value of the j^{th} bit—is straightforwardly done by executing the oracle a few times and performing single qubit measurements. Hence for our purposes we assume that the coefficients $\bar{A}_j$ and thus also $\|\bar{A}\|_2$ are empirically known to sufficient accuracy; we analytically calculate or estimate the scaling of this quantity for various distributions in appendix A.

Yet even if we know the coefficients $\bar{A}_j$ to some precision by an initial sampling procedure, how difficult is it to prepare this new initial state? We first note that the number of *coefficients* of $\left|\bar{A}\right\rangle$ equals the number of *bits of precision* of $|A\rangle$, i.e. g; as such, if e.g. $g = \mathrm{O}(\log N)$, the cost of preparing $\left|\bar{A}\right\rangle$ is negligible (i.e. an at most polylogarithmic overhead in N) in comparison to the speedup of using this optimized bootstrapping routine, which as per table 2 can be polynomial or even exponential in N. In turn, the precision requirements on the coefficients $\bar{A}_j$ are similarly mild: in the step prior to eq. (18), an approximate average state $\left|\bar{B}\right\rangle$ would introduce a relative error in our runtime estimate L' of magnitude

$$\frac{L'_{\bar{B}}}{L'_{\bar{A}}} = \frac{\|\bar{A}\|_2 \|\bar{B}\|_2}{\left\langle \bar{A} \middle| \bar{B} \right\rangle}$$

which shows that knowledge of the coefficients $\bar{A}_j$ to g bits of precision is sufficient for a slowdown of at most $\mathrm{O}(1)$.

We summarise the obtainable speedups when using optimised initial states for a series of amplitude vectors in table 2. Particularly noteworthy is that states not even *that* close to uniform—such as for a triangular set of coefficients where $\alpha_i \propto i$ with a trivial oracle unitary $\mathbf{U}_{\mathrm{amp}} = \mathbb{1}$)—can still be simple to load; they require $\mathrm{O}(1)$ amplitude amplification rounds . Furthermore, if the coefficients themselves are unknown (e.g. for a random oracle) but at least their expected distribution is

known, similar statements of the runtime *in expectation* can be found. For instance for coefficients sampled uniformly at random from the unit interval (modulo normalisation) we empirically find a O(1) scaling for state loading as well.

We note that the number of amplitude amplification rounds given by the L'-times from table 2 for the powerlaw distribution appears to match up with the "search with advice" protocol by Montanaro [Mon09, Prop. 3.4], in the sense that if we used our state preparation protocol to prepare an advice state, then the "search with advice" runtime T empirically satisfies $T \times L' = \mathrm{O}(\sqrt{N})$.

4 Generalised State Loading

As already mentioned in the introduction, we note that our black-box state loading technique really loads a state

$$|\psi\rangle = \sum_{i=0}^{N-1} \left(\sum_{j=1}^{g} b_{ij} w_j \right) |i\rangle$$

for some boolean matrix $\mathbf{B} = (b_{ij})$ and weight vector $\mathbf{w} = (w_j)$, where so far we simply assumed that $w_j = 2^{-j}$—which is ideal for loading a binary representation of the coefficients $\alpha_i \equiv \sum_j b_{ij} w_j = \sum_j A_{ij} 2^{-j}$. Yet we can consider the more general case where the w_j are arbitrary weights, e.g. distances in a weighted graph for producing a state to sample from a Travelling Salesman instance, or any other task where fixed scores w_j are assigned (or not assigned) to individual items i.

The state loading procedure is the same as in section 2.1 but for a change in the amplitude gradient state created with a unitary $\mathbf{G}$, which we replace with a unitary

$$\mathbf{W} |0\rangle = \sum_{j=0}^{g-1} \sqrt{w_j} |j\rangle .$$

Naturally, optimised bootstrapping as explained in section 3 can be applied in this context as well.

5 Conclusion

We derive runtime bounds for this optimised state loading protocol, and evaluate them analytically—or empirically, where an analytic evaluation is difficult—for a set of widely-used distributions. We find significant speedups as compared to agnostic black box state loading: e.g. if the amplitudes follow a powerlaw distribution $\propto r^{-k}$ over 10^6 elements, agnostic black box state loading would require $\sim 10^3$ amplification rounds, irrespective of the powerlaw's falloff exponent k. For $k = 2$—where most of the probability mass is concentrated on a few elements—we still only require ~ 440 amplification rounds; if the exponent was $1/2$, it would be only 9.

These speedups in terms of rounds of required amplitude amplification are possible even if we utilize the same oracle as [San+19; GR02]. Yet if we have access to an oracle that can query individual bits of the amplitudes to be loaded, obtaining amplitudes with 64 bits of precision is possible with 6 ancillas; whereas [San+19]'s protocol would require 66 (cf. table 1). We again emphasize that this comparison between algorithms that query fundamentally different oracles is not meant to imply that the oracles themselves are comparable, it is a useful comparison in terms of the overall task to be achieved (loading amplitudes in a black box setting), and potentially interesting as a comparison between the usefulness of various oracle variants themselves.

While our technique is only described for non-negative amplitudes, we expect that they can be extended to negative and complex amplitudes (or those given in a different number representation, such as polar coordinates) in a similar fashion to Sanders et al., or by utilizing a phase gradient

state in conjunction with an oracle can query magnitude and phase of the amplitudes to be loaded directly.

There are further optimisations one could think of. If, for instance, the amplitudes are all from a fixed set of numbers $\alpha_i \in S$, then it is conceivable that a more efficient number representation than binary can be derived. Similar to the generalised state loading described in section 4, such a representation would likely improve the runtime further, and require even fewer ancillas. Finally, an interesting question to pursue would be in what concrete context the generalised black box state loading protocol can be employed.

Acknowledgements

We would like to thank Sathya Subramanian, Felix Leditzky and Yuval Sanders for helpful discussions, as well as the reviewers at Quantum for very useful feedback. J. B. is supported by the Draper's Research Fellowship at Pembroke College.

References

[Gro00] Lov K. Grover. "Synthesis of Quantum Superpositions by Quantum Computation". *Physical Review Letters* 85.6 (Aug. 2000), pp. 1334–1337.

[San+19] Yuval R. Sanders, Guang Hao Low, Artur Scherer, and Dominic W. Berry. "Black-Box Quantum State Preparation without Arithmetic". *Physical Review Letters* 122.2 (Jan. 2019), p. 020502.

[HHL09] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". *Physical Review Letters* 103.15 (Oct. 2009), p. 150502. arXiv: 0811.3171.

[Kem03] Julia Kempe. "Quantum random walks - an introductory overview". *Contemporary Physics* 44.4 (Mar. 2003), pp. 307–327. arXiv: 0303081 [quant-ph].

[San08] Miklos Santha. "Quantum Walk Based Search Algorithms". In: *Theory and Applications of Models of Computation*. Ed. by Manindra Agrawal, Dingzhu Du, Zhenhua Duan, and Angsheng Li. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 31–46.

[BC09] Dominic W. Berry and Andrew M. Childs. "Black-box Hamiltonian simulation and unitary implementation" (Oct. 2009). arXiv: 0910.4157.

[Bra+17] Fernando G. S. L. Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M. Svore, and Xiaodi Wu. "Quantum SDP Solvers: Large Speed-ups, Optimality, and Applications to Quantum Learning". In: *ICALP 2019*. Oct. 2017. arXiv: 1710.02581.

[Bra+19] Sergey Bravyi, Alexander Kliesch, Robert Koenig, and Eugene Tang. "Obstacles to State Preparation and Variational Optimization from Symmetry Protection" (Oct. 2019). arXiv: 1910.08980.

[BCK15] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. "Hamiltonian simulation with nearly optimal dependence on all parameters" (Jan. 2015). arXiv: 1501.01715.

[Cod+12] N Cody Jones, James D. Whitfield, Peter L. McMahon, Man-Hong Yung, Rodney Van Meter, Alán Aspuru-Guzik, and Yoshihisa Yamamoto. "Faster quantum chemistry simulation on fault-tolerant quantum computers". *New Journal of Physics* 14.11 (Nov. 2012), p. 115023. arXiv: 1204.0567.

[SS06] Andrei N. Soklakov and Rüdiger Schack. "Efficient state preparation for a register of quantum bits". *Physical Review A* 73.1 (Jan. 2006), p. 012307.

[PB11] Martin Plesch and Časlav Brukner. "Quantum-state preparation with universal gate decompositions". *Physical Review A* 83.3 (Mar. 2011), p. 032302. arXiv: 1003.5760.

[Mot+04] Mikko Mottonen, Juha J. Vartiainen, Ville Bergholm, and Martti M. Salomaa. "Transformation of quantum states using uniformly controlled rotations". *Quant. Inf. Comp.* 5 (July 2004), p. 467. arXiv: 0407010 [quant-ph].

[Ara+20] Israel F. Araujo, Daniel K. Park, Francesco Petruccione, and Adenilton J. da Silva. "A divide-and-conquer algorithm for quantum state preparation" (Aug. 2020). arXiv: 2008.01511.

[GR02] Lov Grover and Terry Rudolph. "Creating superpositions that correspond to efficiently integrable probability distributions" (Aug. 2002). arXiv: 0208112 [quant-ph].

[Chi09] Andrew M. Childs. "On the Relationship Between Continuous- and Discrete-Time Quantum Walk". *Communications in Mathematical Physics* 294.2 (Oct. 2009), pp. 581–603.

[ZLW19] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. "Quantum Generative Adversarial Networks for learning and loading random distributions". *npj Quantum Information* (2019). arXiv: 1904.00043.

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge: Cambridge University Press, 2010, p. 676.

[YLC14] Theodore J. Yoder, Guang Hao Low, and Isaac L. Chuang. "Fixed-Point Quantum Search with an Optimal Number of Queries". *Physical Review Letters* 113.21 (Nov. 2014), p. 210501.

[Cuc+04] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. "A new quantum ripple-carry addition circuit" (Oct. 2004). arXiv: 0410184 [quant-ph].

[Gid18] Craig Gidney. "Halving the cost of quantum addition". *Quantum* 2 (June 2018), p. 74. arXiv: 1709.06648.

[He+17] Yong He, Ming-Xing Luo, E. Zhang, Hong-Ke Wang, and Xiao-Feng Wang. "Decompositions of n-qubit Toffoli Gates with Linear Circuit Complexity". *International Journal of Theoretical Physics* 56.7 (July 2017), pp. 2350–2361.

[SD96] John A. Smolin and David P. DiVincenzo. "Five two-bit quantum gates are sufficient to implement the quantum Fredkin gate". *Physical Review A* 53.4 (Apr. 1996), pp. 2855–2856.

[Goo+19] Quantum AI Lab Google et al. "Quantum supremacy using a programmable superconducting processor". *Nature* 574.7779 (Oct. 2019), pp. 505–510.

[Mon09] Ashley Montanaro. "Quantum search with advice" (Aug. 2009), pp. 1–14. arXiv: 0908.3066.

A Optimized Initial State Calculations

We present calculations for the speedups presented in table 2, focusing on the dependence of L' given in eq. (20) in N, i.e.

$$L' \sim \sqrt{N}\frac{\|\alpha\|_1}{\|\bar{A}\|_2}.$$

Delta Distribution. A single element is marked; hence $\|\alpha\|_1 = \sum_i \alpha_{i=0}^{N-1} = 1$. With an optimized state loading protocol we first need to determine the average bit weights $\bar{A}_j$ as defined just before eq. (18), i.e.

$$\bar{A}_j = 2^{-j/2}\sum_{i=0}^{N-1} A_{ij}.$$

As A_{i0} denotes the most significant bit, i.e. the 1/2's, as laid out at the start of section 2.1, we can assume that for the single marked element i' we have $A_{i'j} = 1$ for all $j = 1, \ldots, g$, and $A_{ij} = 0$ for all $i \neq i'$. Then $\bar{A}_j = 2^{-j/2}$, and thus

$$1 - 2^{-g} \leq \|\bar{A}\|_2^2 = \sum_{j=1}^{g} 2^{-j} \leq 1$$

By eq. (20) we get $L'_1 = \mathrm{O}(\sqrt{N})$.

Uniform Distribution. All elements are marked; hence $\alpha_i = 1/\sqrt{N}$, and $\|\alpha\|_1 = \sum_i \alpha_i = N/\sqrt{N} = \sqrt{N}$. For the discrete case we assume there exists one k such that $2^{-k} = 1/\sqrt{N}$. For this k we then have $A_{ik} = 1$ and $A_{ij} = 0$ for all $j \neq k$, and for all i. Thus $\bar{A}_k = 2^{-k/2} \times N = N^{3/4}$ and $\bar{A}_j = 0$ for all $j \neq k$. Hence $\|\bar{A}\|_2 = N^{3/4}$. By eq. (20) we have $L' = \mathrm{O}(N^{1/4})$.

This is not yet ideal; we want to obtain a $\mathrm{O}(1)$ scaling. By choosing a different number representation in which we obmit all higher-order bits $j < k$, we obtain $\bar{A}_1 = N/\sqrt{2}$ and $\bar{A}_j = 0$ for $j \neq k$; thus $\|\bar{A}\|_2 \sim N$, and the constant runtime follows.

It is worth noting that this particular shortcut is a variant of using the more generalised state loading protocol from section 4, with the weights corresponding to an optimized choice of the distribution's bit representation.

Triangle Distribution. We have $\alpha_i \propto i$ for $i = 0, \ldots, N-1$, which with normalisation under the 2-norm yields

$$\alpha_i = i/\sqrt{N(N-1)(2N-1)/6}.$$

This means

$$\|\alpha\|_1 = \sqrt{\frac{3}{2}}\frac{N(N-1)}{\sqrt{N(N-1)(2N-1)}} \sim \sqrt{\frac{3}{4}}\frac{N^2}{N^{3/2}} = \sqrt{\frac{3}{4}}\sqrt{N}.$$

We assume for simplicity that N and the bit representation is chosen such that

$$A_0 = \texttt{0.0}\cdots\texttt{000},\ A_1 = \texttt{0.0}\cdots\texttt{001},\ A_2 = \texttt{0.0}\cdots\texttt{010},\ \ldots,\ A_{N-1} = \texttt{0.1}\cdots\texttt{111}.$$

This means each bit is equally likely, and is set to one for precisely half of the N amplitudes; it immediately follows that $\bar{A}_j = 2^{-j/2}N/2$, and thus

$$\|\bar{A}\|_2 = \frac{N}{2}\sqrt{\sum_{j=1}^{g} 2^{-j}} = \sqrt{1-2^{-g}}\frac{N}{2}.$$

It follows that $L' = \mathrm{O}(1)$.

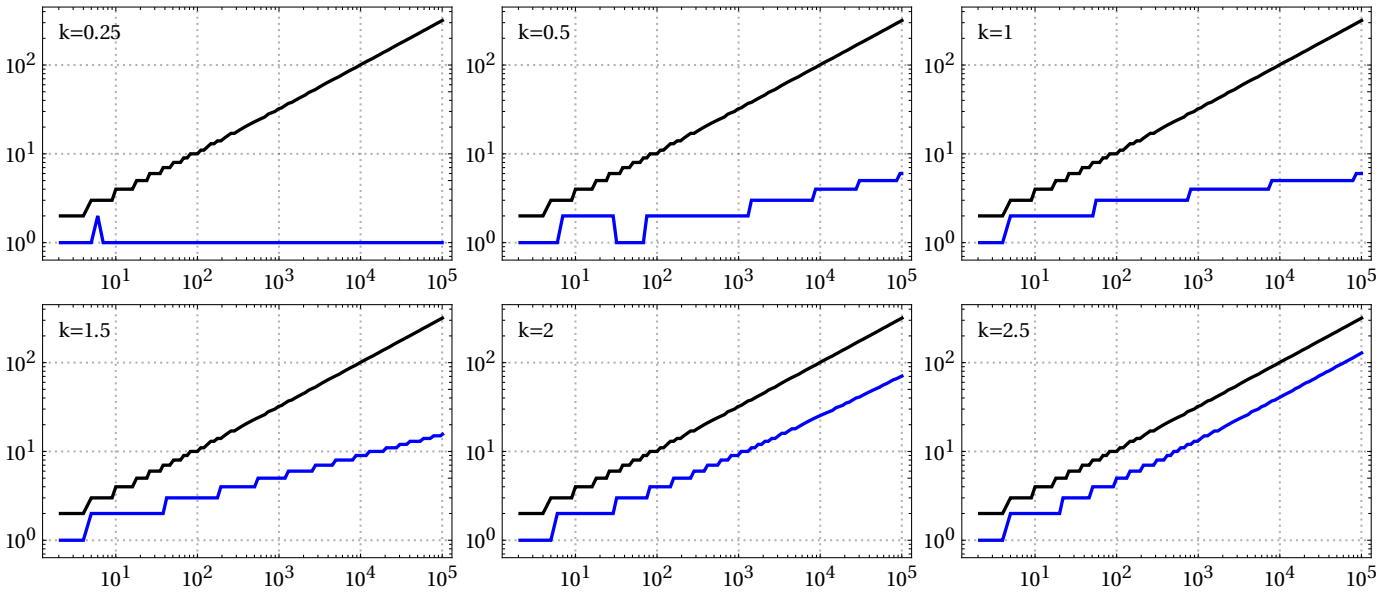

Figure 3: Runtime L (black, from eq. (14)) and L' (blue, from eq. (20)) on vertical axis, evaluated empirically for a powerlaw distribution with various choices of the falloff exponent k, and various element counts N on the horizontal axis. The kinks where the runtime decreases for increasing N emerge from cases where we shifted the highest-significance bit to more effectively represent the distribution's dynamic range, as done and explained in the case of the uniform distribution.

Powerlaw Distribution. A powerlaw distribution is given by weights $\propto r^{-k}$, for $k \in (0, \infty)$, and where k indicates the rank of the element (if sorted). As our amplitude vector α is normalised by the 2-norm, we will assume the task to be to load the amplitudes

$$\alpha_r = \frac{r^{-k}}{H_{N,2k}^{1/2}} \quad \text{for } r = 1, \ldots, N$$

and where the normalisation is given by the Harmonic number $H_{N,k} \coloneqq \sum_{r=1}^{N} r^{-k}$; then $\|\alpha\|_2 = 1$ as can be easily verified. Fig. 3 shows the resulting runtimes, for various powerlaw falloff choices.

Normal Distribution. A normal distribution is given by weights $\propto \exp(-x^2/2\sigma^2)$, for $x \in \mathbb{R}$, and where $\sigma > 0$ denotes the standard deviation. In our discrete case, we simply assume that α_x is proportional to this weight, normalised such that $\|\alpha\|_2 = 1$. Fig. 4 shows the resulting runtimes, for various standard deviation choices.

The scaling suggests that if the standard deviation is large in the context of the number of samples, then the runtime is $\mathrm{O}(1)$; otherwise $\mathrm{O}(\sqrt{N/\sigma})$.

Random Distribution. In case the distribution we wish to load is itself randomly sampled uniformly from the interval $[0, 1]$—i.e. not with uniform weights, but where the α_i are random samples from $[0, 1]$, and the overall vector normalised—one can calculate the runtime (in expectation). If $X_i \sim \mathrm{unif}(0, 1)$ are uniform iid random variables, we have

$$\|\alpha\|_1 = \frac{\sum_{i=0}^{N-1} \alpha_i}{\sqrt{\sum_{i=0}^{N-1} \alpha_i}} = \frac{\sum_i \mathbb{E}(X_i)}{\mathbb{E}\left[\sqrt{\sum_i X_i^2}\right]} \leq \frac{N/2}{\sqrt{N}} = \sqrt{N}/2.$$

We can check empirically how $\|\bar{A}\|_2$ scales; this is shown in fig. 5, and the scaling suggests that $\|\bar{A}\|_2 = \Omega(N)$. As a result, by eq. (20), we have $L' = \mathrm{O}(1)$.

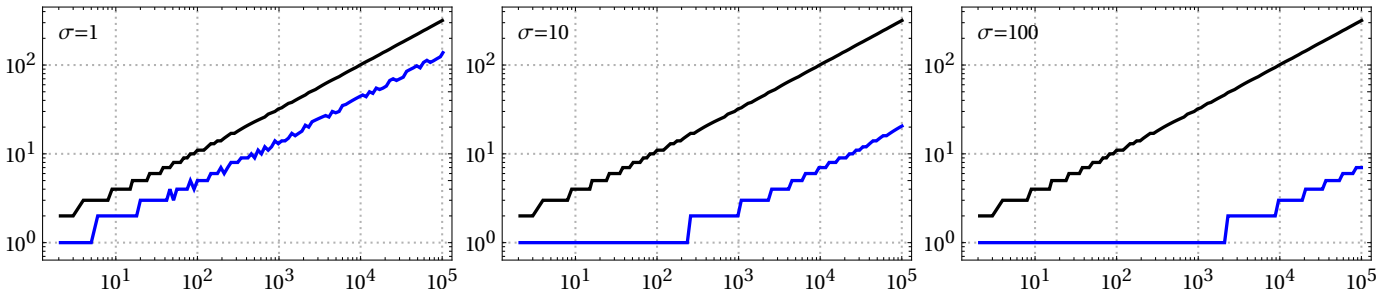

Figure 4: Runtime L (black, from eq. (14)) and L' (blue, from eq. (20)) on vertical axis, evaluated empirically for a normal distribution with various choices of standard deviation σ, and various element counts N on the horizontal axis. The kinks where the runtime decreases for increasing N emerge from cases where we shifted the highest-significance bit to more effectively represent the distribution's dynamic range, as done and explained in the case of the uniform distribution.

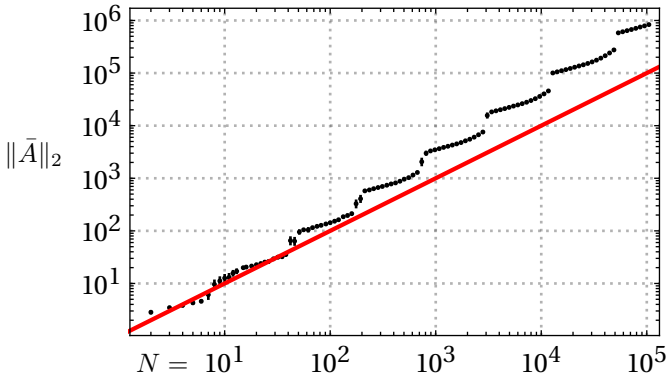

Figure 5: 2-norm of average bit weight $\|\bar{A}\|_2$ (black) for a random distribution, where amplitudes are drawn uniformly at random from $[0, 1]$, and then normalized. As reference in red is the diagonal N, indicating that $\|\bar{A}\|_2 = \Omega(N)$.

Sin Distribution. For discrete-time random walks, and as mentioned in the introduction, an interesting initial state to prepare is when the $\alpha_i = \sqrt{2/(n+1)} \times \sin(\pi(i+1)/(N+1))$. A similar argument to the triangle distribution case shows $L' = \mathrm{O}(1)$.