Quantum gradient descent and Newton's method for constrained polynomial optimization

PAPER • OPEN ACCESS

Quantum gradient descent and Newton's method for constrained polynomial optimization

New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

OPEN ACCESS

RECEIVED
16 May 2019

ACCEPTED FOR PUBLICATION
18 June 2019

PUBLISHED
17 July 2019

PAPER

Quantum gradient descent and Newton's method for constrained polynomial optimization

Patrick Rebentrost[1,2], **Maria Schuld**[3,4,10], **Leonard Wossnig**[5,6,7], **Francesco Petruccione**[3,8] and **Seth Lloyd**[2,9]

[1] Centre for Quantum Technologies, National University of Singapore, 3 Science Drive 2, Singapore 117543, Singapore
[2] Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139, United States of America
[3] Quantum Research Group, School of Chemistry and Physics, University of KwaZulu-Natal, Durban 4000, South Africa
[4] Xanadu, 777 Bay Street, Toronto, M5B 2H7, Canada
[5] Institute for Theoretical Physics, ETH Zurich, 8093 Zurich, Switzerland
[6] Department of Materials, University of Oxford, Parks Road, Oxford OX1 3PH, United Kingdom
[7] Department of Computer Science, University College London, Gower Street, London WC1E 7JE, United Kingdom
[8] National Institute for Theoretical Physics, KwaZulu-Natal, South Africa
[9] Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, United States of America
[10] Author to whom any correspondence should be addressed.

E-mail: cqtfpr@nus.edu.sg, pr@patrickre.com and schuld@ukzn.ac.za

Keywords: quantum optimization, quantum computing, density matrix exponentiation

Abstract

Optimization problems in disciplines such as machine learning are commonly solved with iterative methods. Gradient descent algorithms find local minima by moving along the direction of steepest descent while Newton's method takes into account curvature information and thereby often improves convergence. Here, we develop quantum versions of these iterative optimization algorithms and apply them to polynomial optimization with a unit norm constraint. In each step, multiple copies of the current candidate are used to improve the candidate using quantum phase estimation, an adapted quantum state exponentiation scheme, as well as quantum matrix multiplications and inversions. The required operations perform polylogarithmically in the dimension of the solution vector and exponentially in the number of iterations. Therefore, the quantum algorithm can be useful for high-dimensional problems where a small number of iterations is sufficient.

1. Introduction

Optimization plays a vital role in various fields. In machine learning and artificial intelligence, common techniques such as regression, support vectors machines, and neural networks rely on optimization. In these cases the objective function to optimize is often a least-squares loss or error function $f(\mathbf{x})$ that takes a vector-valued input to a scalar-valued output [1]. For strictly convex functions on a compact convex set, there exists a unique global minimum, and, in the case of equality-constrained quadratic programming, solving the optimization problem reduces to a matrix inversion problem. In machine learning, one often deals with either convex objective functions beyond quadratic programming, or non-convex objective functions with multiple minima. In these situations, no single-shot solution is known in general and one usually has to resort to iterative search in the landscape defined by the objective function.

One popular approach focuses on gradient descent methods such as the famous backpropagation algorithm in the training of neural networks. Gradient descent finds a local minimum starting from an initial guess by iteratively proceeding along the negative gradient of the objective function. Because it only takes into account the first derivatives of $f(\mathbf{x})$, gradient descent may involve many steps in cases when the problem has an unfortunate landscape[11]. For such objective functions, second-order methods, which model the local curvature and correct the gradient step size, have been shown to perform well [2]. One such method, the so-called Newton's method,

[11] For example where it features long and narrow valleys. A famous example is the Rosenbrock function, for which gradient descent fails completely.

multiplies the inverse Hessian to the gradient of the function. By taking into account the curvature information in such a manner, the number of steps required to find the minimum often greatly reduces at the cost of computing and inverting the matrix of the second derivatives of the function with respect to all coordinates. Once the method arrives in the vicinity of a minimum, the algorithm enters a realm of *quadratic convergence*, where the number of correct bits in the solution doubles with every step [3, 4].

As quantum computation becomes more realistic and ventures into the field of machine learning, it is worthwhile to consider in what way optimization algorithms can be translated into a quantum computational framework. Optimization has been considered in various implementation proposals of quantum computing [5, 6]. The adiabatic quantum computing paradigm [7] and its famous sibling, quantum annealing, are strategies to find the ground state of a Hamiltonian and can therefore be understood as 'analog' algorithms for optimization problems. The first commercial implementation of quantum annealing, the D-Wave machine, solves certain quadratic unconstrained optimization problems and has been tested for machine learning applications such as classification [8, 9] and sampling for the training of Boltzmann machines [10]. In the gate model of quantum computation, quadratic optimization problems deriving from machine learning tasks such as least-squares regression [11, 12] and the least-squares support vector machine [13] were tackled with the quantum matrix inversion technique [14], demonstrating the potential for speedups for such single-shot solutions under certain conditions. Variational methods that use classical optimization while computing an objective function on a quantum computer have become popular methods targeting near-term devices with limited coherence times [15, 16].

In this work, we provide quantum algorithms for iterative optimization, specifically the gradient descent and Newton's methods. We thereby extend the quantum machine learning literature by techniques that can be used in non-quadratic convex or even non-convex optimization problems. The main idea is that at each step we take multiple copies of a quantum state $|\mathbf{x}^{(t)}\rangle$ to produce multiple copies of another quantum state $|\mathbf{x}^{(t+1)}\rangle$ by considering the gradient vector and the Hessian matrix of the objective function. Depending on the step size taken, this quantum state improves the objective function.

We consider optimization for the special case of polynomials with a normalization constraint. The class of polynomials we discuss in detail for optimization contains homogeneous polynomials of even degree over large-dimensional spaces with a relatively small number of monomials. We also sketch the polynomial optimization with a relatively small number of inhomogeneous terms added. We show how to make the gradient of the objective function and the Hessian matrix operationally available in the quantum computation using techniques such as quantum state exponentiation [17]. The objective function is assumed to be given via oracles providing access to the coefficients of the polynomial and allowing black-box Hamiltonian simulation as described in [18].

Since at each step we consume multiple copies of the current solution to prepare a single copy of the next step, the algorithms scale exponentially in the number of steps performed. While this exponential scaling precludes the use of our algorithms for optimizations that require many iterations, it is acceptable in cases when only a few steps are needed to arrive at a reasonably good solution, especially in the case of the Newton method, or when performing a local search. We note that the computation of gradients on a quantum computer has been investigated before, for a setting in which the coordinates of the input vectors to the function are encoded as binary numbers rather than as the amplitudes of a quantum system [19]. We also note subsequent work on quantum gradient descent in [20] and [21].

2. Problem statement

Gradient descent is formulated as follows. Let $f\colon \mathbb{R}^N \to \mathbb{R}$ be the objective function one intends to minimize. Given an initial point $\mathbf{x}^{(0)} \in \mathbb{R}^N$, one iteratively updates this point using the steepest descent of the objective function at the current point,

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta \, \nabla f(\mathbf{x}^{(t)}), \tag{1}$$

where $\eta > 0$ is a hyperparameter (called the learning rate in a machine learning context) which may in general be step-dependent. Newton's method extends this strategy by taking into account information about the curvature, i.e. the second derivatives of the objective function, in every step. The iterative update therefore includes the Hessian matrix $\mathbf{H}$ of the objective function,

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta \mathbf{H}^{-1} \nabla f(\mathbf{x}^{(t)}), \tag{2}$$

with the entries $H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$ evaluated at the current point $\mathbf{x}^{(t)}$.

These iterative methods are in principle applicable to any sufficiently smooth function. In this work, we restrict ourselves to the optimization of multivariate homogeneous polynomials of even degree and under

spherical constraints, where the polynomials only consist of a small number of terms (a property which we refer to as 'sparse'). We also discuss the extension to a small number of *inhomogeneities*.

2.1. Sparse, even homogenous polynomials

The *homogeneous* objective function we seek to minimize here is a polynomial of degree $2p$ defined over $\mathbf{x} \in \mathbb{R}^N$,

$$f(\mathbf{x}) = \frac{1}{2} \sum_{i_1,\ldots,i_{2p}=1}^{N} A_{i_1 \ldots i_{2p}} x_{i_1} \ldots x_{i_{2p}}, \tag{3}$$

with the N^{2p} coefficients $A_{i_1 \ldots i_{2p}} \in \mathbb{R}$ and $\mathbf{x} = (x_1, \ldots, x_N)^T$. As we will see, the number of non-zero coefficients will appear in the final resource analysis of the quantum algorithm. More precisely, the algorithm will depend on the 'sparsity' of a matrix $\mathbf{A}$ constructed from the coefficients $A_{i_1 \ldots i_{2p}}$, where sparsity refers to the number of non-zero entries per row and column. By mapping the inputs to a higher-dimensional space constructed from p tensor products of each vector, we can write this function as an *algebraic form*,

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \otimes \cdots \otimes \mathbf{x}^T \mathbf{A}\, \mathbf{x} \otimes \cdots \otimes \mathbf{x}. \tag{4}$$

The coefficients become entries of an order-p tensor in $\mathbb{R}^{N\times N} \otimes \cdots \otimes \mathbb{R}^{N\times N}$, or equivalently entries of an $N^p \times N^p$ dimensional matrix $\mathbf{A}$. Equations (3) and (4) describe a homogeneous polynomial of even degree. For the more familiar case of $p = 1$, the objective function reduces to $f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$ and one obtains a common quadratic optimization problem. For $p = 2$ and $N = 2$, the two-dimensional input $\mathbf{x} = (x_1, x_2)^T$ results in $\mathbf{x} \otimes \mathbf{x} = (x_1^2, x_1x_2, x_2x_1, x_2^2)^T$, which contains the polynomial terms up to quadratic, and $\mathbf{A}$ is of size 4×4. In this work, we use the operator norm $\|\cdot\| = \max_j |\lambda_j(\cdot)|$, where λ_j are the eigenvalues of the diagonalizable matrix under consideration. Let $\Lambda_A \geqslant 1$ be given such that the norm of $\mathbf{A}$ is $\|\mathbf{A}\| \leqslant \Lambda_A$.

It will be helpful to formally decompose $\mathbf{A}$ into a sum of tensor products of matrices acting on each of the p vector spaces that comprise the full vector space,

$$\mathbf{A} = \sum_{\alpha=1}^{K} \mathbf{A}_1^\alpha \otimes \cdots \otimes \mathbf{A}_p^\alpha, \tag{5}$$

where each $\mathbf{A}_i^\alpha$ is a $N \times N$ matrix for $i = 1, \ldots, p$ and K is the number of terms in the sum required to specify $\mathbf{A}$. Since the quadratic form remains the same by $\mathbf{x}^T\mathbf{A}_i^\alpha\mathbf{x} = \mathbf{x}^T(\mathbf{A}_i^\alpha + \mathbf{A}_i^{\alpha,T})\mathbf{x}/2$ we can assume without loss of generality that the $\mathbf{A}_i^\alpha$ are symmetric and also that $\mathbf{A}$ is symmetric. Note that the representation of equation (5) is useful to simplify the computation of the gradient and the Hessian matrix of $f(\mathbf{x})$. However, our quantum algorithm does not explicitly require this tensor decomposition of $\mathbf{A}$, which may be hard to compute in general. We only require access to the matrix elements of $\mathbf{A}$ for the use in standard Hamiltonian simulation methods [18] (see the data input discussion below). As mentioned above, we consider general but sparse matrices $\mathbf{A}$, where the sparsity requirement arises from the quantum simulation methods. A sparse matrix $\mathbf{A}$ represents a polynomial with a relatively small number of monomials.

For the gradient descent and Newton's methods, we require expressions for the gradient and the Hessian matrix of the objective function. In the tensor formulation of equation (5), the gradient of the objective function at point $\mathbf{x}$ can be written as

$$\nabla f(\mathbf{x}) = \sum_{\alpha=1}^{K} \sum_{j=1}^{p} \left(\prod_{\substack{i=1 \\ i\neq j}}^{p} \mathbf{x}^T\mathbf{A}_i^\alpha\mathbf{x} \right) \mathbf{A}_j^\alpha\, \mathbf{x} =: \mathbf{D}(\mathbf{x})\mathbf{x}, \tag{6}$$

which can be interpreted as an operator $\mathbf{D} \equiv \mathbf{D}(\mathbf{x})$ which is a sum of matrices $\mathbf{A}_j^\alpha$ with $\mathbf{x}$-dependent coefficients $\prod_{i\neq j} \mathbf{x}^T\mathbf{A}_i^\alpha\mathbf{x}$, applied to a single vector $\mathbf{x}$. Note that of course the ordering of the scalar factors in the product $\prod_{\substack{i=1 \\ i\neq j}}^{p} \mathbf{x}^T\mathbf{A}_i^\alpha\mathbf{x}$ is unimportant. In addition, for the norm of the operator $\mathbf{D}$, note that with $|\mathbf{x}| \leqslant 1$ we have $\mathbf{x}^T\mathbf{D}(\mathbf{x})\mathbf{x} \leqslant \max_{|\mathbf{y}|\leqslant 1} \mathbf{y}^T\mathbf{D}(\mathbf{x})\mathbf{y} \leqslant p \max_{|\mathbf{y}|\leqslant 1} (\mathbf{y} \otimes \ldots \otimes \mathbf{y})^T\, \mathbf{A}\mathbf{y} \otimes \ldots \otimes \mathbf{y} \leqslant p\Lambda_A$, where $\|\mathbf{A}\| \leqslant \Lambda_A$. Thus, we can define Λ_A such that $\|\mathbf{D}\| \leqslant p\Lambda_A$.

In a similar fashion, the Hessian matrix at the same point reads,

$$\mathbf{H}(f(\mathbf{x})) = 2 \sum_{\alpha=1}^{K} \sum_{\substack{j,k=1 \\ j\neq k}}^{p} \prod_{\substack{i=1 \\ i\neq j,k}}^{p} (\mathbf{x}^T\mathbf{A}_i^\alpha\mathbf{x})\, \mathbf{A}_k^\alpha\mathbf{x}\, \mathbf{x}^T\mathbf{A}_j^\alpha + \mathbf{D} =: \mathbf{H}_1 + \mathbf{D}, \tag{7}$$

defining the operator $\mathbf{H}_1$. For the matrix norm of the operator $\mathbf{H}$, note that with $|\mathbf{x}| \leqslant 1$ we have $\mathbf{x}^T\mathbf{H}(\mathbf{x})\mathbf{x} \leqslant \max_{|\mathbf{y}|\leqslant 1} \mathbf{y}^T\, \mathbf{H}(\mathbf{x})\mathbf{y} \leqslant 2p^2 \max_{|\mathbf{y}|\leqslant 1} (\mathbf{y} \otimes \ldots \otimes \mathbf{y})^T\, \mathbf{A}\mathbf{y} \otimes \ldots \otimes \mathbf{y} + p\Lambda_A \leqslant 2p^2\Lambda_A$. Thus we can redefine Λ_A such that $\|\mathbf{H}\| \leqslant p^2\Lambda_A$. The core of the quantum algorithms for gradient descent and Newton's method will be to implement the matrices $\mathbf{D}$ and $\mathbf{H}$ as quantum operators acting on the current state and successively shifting the current state towards the desired minimum.

2.2. Spherical constraints

Since in this work we represent vectors $\mathbf{x}$ as quantum states, the quantum algorithm naturally produces normalized vectors with $\mathbf{x}^T\mathbf{x} = 1$, thereby implementing a constraint known in the optimization literature as a *spherical constraint*. Applications of such optimization problems appear in image and signal processing, biomedical engineering, speech recognition and quantum mechanics [22].

In addition, we include further standard assumptions [4]. We assume an initial guess $\mathbf{x}_0$ reasonably close to the constraint local minimum. We assume that the Hessian of the polynomial in equation (7) in the vicinity of the solution is positive semidefinite. We also note that every polynomial is smooth and Lipschitz continuous on the unit ball. These properties guarantee saddle-point free optimization and monotonic convergence to the minimum. The problem we attempt to solve is therefore defined as follows.

Problem 1. Let $f(x)$ be a homogeneous polynomial of even degree $2p$ with $p \in \mathbb{Z} > 0$ as in equation (4). Let the matrix $\mathbf{A}$ defining $f(x)$ be symmetric and have sparsity s_A, defined as the number of non-zero matrix elements in each row and column. In addition, let $\Lambda_A \geqslant 1$ be given such that the matrix norm of $\mathbf{A}$ is $\|\mathbf{A}\| \leqslant \Lambda_A$. Starting with an initial guess $\mathbf{x}_0$, attempt to solve the problem

$$\min_{\mathbf{x}} f(\mathbf{x}),$$

subject to the constraint $\mathbf{x}^T\mathbf{x} = 1$ by finding a local minimum $\mathbf{x}^*$. We assume that the Hessian is positive semidefinite at the solution, i.e. $\mathbf{H}(f(\mathbf{x}^*)) \geqslant 0$, and that the initial guess is sufficiently close to the solution.

A well-known adaptation of gradient descent for such constrained problems is *projected gradient descent* [23, 24], where after each iteration the current solution is projected onto the feasible set (here corresponding to a renormalization to $\mathbf{x}^T\mathbf{x} = 1$). It turns out that our quantum algorithms naturally follow this procedure and the renormalization is realized in each update of the quantum state. We therefore obtain the general convergence properties of the projected gradient descent and projected Newton's methods. Note that for the simple case of the quadratic function $\mathbf{x}^T\mathbf{A}\mathbf{x}$, i.e. for $p = 1$, gradient descent with a unit norm constraint finds one of the eigenstates of $\mathbf{A}$.

Although the choice of the objective function allows for an elegant implementation of the operators $\mathbf{D}$ and $\mathbf{H}^{-1}$ by means of quantum information processing, it is in some cases not suited for Newton's method. For example, if $p = 1$ the objective function reduces to a quadratic form and $\mathbf{H}^{-1}\nabla f(\mathbf{x}) = \mathbf{D}^{-1}\mathbf{D}\mathbf{x} = \mathbf{x}$. The direction of search is consequently perpendicular to the unit sphere and Newton's method does not update the initial guess at all (see figure 2). For this reason and, more generally, to increase the class of functions that can be optimized, it is interesting to include inhomogeneous terms to the polynomial. For example, a simple linear inhomogeneity can be added by considering the polynomial function $f(\mathbf{x}) + \mathbf{c}^T\mathbf{x}$, where $f(\mathbf{x})$ is the homogeneous part as before and $\mathbf{c}$ is a vector specifying the inhomogeneous part (see figure 3). We will sketch a method to include such inhomogeneities in section 5.

3. Quantum gradient descent algorithm

3.1. Data input model

To implement a quantum version of the gradient descent algorithm we assume without loss of generality that the dimension N of the vector $\mathbf{x}$ is $N = 2^n$ where n is an integer. A vector not satisfying this condition can always be padded to 2^n as long as the corresponding entries of $\mathbf{A}$ are set to 0. Following previous quantum algorithms [13, 14, 17] we represent the entries of $\mathbf{x} = (x_1, \ldots, x_N)^T$ as the amplitudes of a n-qubit quantum state $|\mathbf{x}\rangle = \frac{1}{|\mathbf{x}|}\sum_{j=1}^{N} x_j|j\rangle$, where $|j\rangle$ is the jth computational basis state.

In this work, we assume the following oracles for the data input. First, we define the oracle for providing copies of the initial state. Let $\mathbf{x}^{(0)} = (x_1^{(0)}, \ldots, x_N^{(0)})^T$ be the initial point we choose as a candidate for finding the minimum $\mathbf{x}^*$ with $\sum_i |x_i^{(0)}|^2 = 1$.

Oracle 1 (Initial state preparation oracle). Assume an oracle that performs the operation $|0\rangle \rightarrow |\mathbf{x}^{(0)}\rangle$ on n qubits.

We assume that the initial quantum state $|\mathbf{x}^{(0)}\rangle$ corresponding to $\mathbf{x}^{(0)}$ via amplitude encoding can be prepared efficiently, either as an output of a quantum algorithm or by preparing the state from quantum random access memory [25–27]. For example, efficient algorithms exist for preparing states corresponding to integrable [28] and bounded [29] distributions. In addition, we assume an oracle for the matrix $\mathbf{A}$ [18].

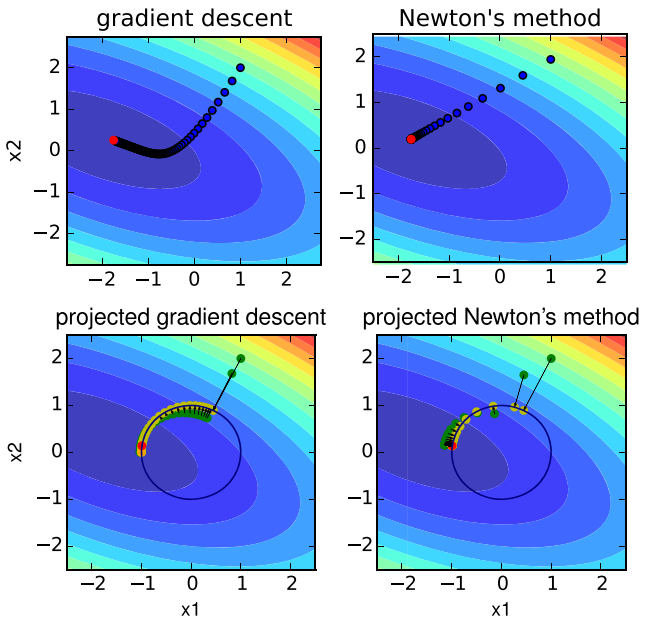

Figure 1. Examples of gradient descent and Newton's method, as well as their projected versions. While gradient descent follows the direction orthogonal to the contour lines to find the minimum marked in red (upper left), Newton's method takes the curvature into account to choose a straighter path (upper right). Projected descent methods under unit sphere constraints renormalize (lower left and right, yellow dots) the solution candidate (green dots) after every update and find the minimum on the feasible set. Considered here is a quadratic objective function $f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{c}^T\mathbf{x}$ with $a_{11} = a_{12} = a_{21} = 0.2$, $a_{22} = 0.6$ and $\mathbf{c} = (0.3,\ 0.2)^T$, as well as step size $\eta = 0.2$ and initial state $\mathbf{x}^{(0)} = (1,\ 2)^T$.

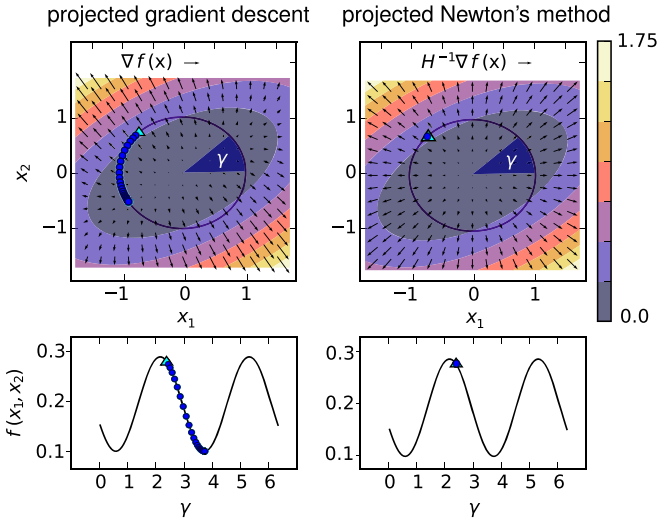

Figure 2. Projected gradient descent and projected Newton's method for quadratic optimization under unit sphere constraints (i.e. the solution is constrained to the circle). Below is the function on the feasible set only, with the angle γ starting from position $(1,\ 0)^T$. The parameters are chosen as $K = 1$, $p = 1$, $N = 2$ and the objective function is $f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$ with $a_{11} = 0.3$, $a_{12} = a_{21} = -0.2$, $a_{22} = 0.5$ and the initial point (light blue triangle) is chosen as $\mathbf{x}^{(0)} = [-0.707,\ 0.707]$. For quadratic forms, Newton's method struggles to find the minimum on the feasible set plotted below, since the field lines of the descent direction $\mathbf{H}^{-1}\nabla f(\mathbf{x})$ are perpendicular to the unit circle. This feature also holds for matrices A that are not positive definite unlike in this example.

Oracle 2 (Polynomial coefficients oracle). Let $j,\ k = 1, \cdots, N^p$ with $N = 2^n$ and the matrix $\mathbf{A}$ be symmetric. Assume an oracle that performs the operation $|j,\ k\rangle|0\rangle \rightarrow |j,\ k\rangle|A_{jk}\rangle$ on $2pn + n_\chi$ qubits where A_{jk} is encoded to accuracy $\chi = 2^{-n_\chi}$.

We assume that the error χ is much smaller than other errors and does not affect the analysis [18]. Note that the indices $j,\ k$ can isomorphically be described by numbers $i_1, \cdots, i_{2p} = 1, \cdots, N$. Thus, the oracle can be equivalently given as $|i_1, \cdots, i_{2p}\rangle|0\rangle \rightarrow |i_1, \cdots, i_{2p}\rangle|A_{i_1, \ldots, i_{2p}}\rangle$. To take advantage of sparsity, we also assume the following oracle, which allows for choosing the non-zero matrix elements.

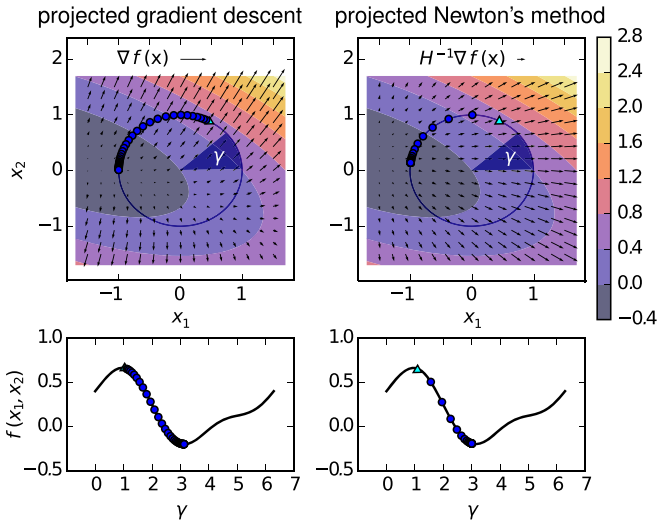

Figure 3. The quantum algorithms can be adapted to optimize polynomials that include inhomogeneities. Here, we consider a quadratic form, where the parameters including the inhomogeneity are the same as in figure 1. The panels are analogous to figure 2. Newton's method arrives at the solution faster than gradient descent.

Oracle 3 (Sparse input oracle). Let $j = 1, \cdots, N^p$ and $l = 1, \cdots, s_A$. Assume an oracle that performs the operation $|j, l\rangle \rightarrow |j, g_A(j, l)\rangle$ on $2pn$ qubits where the efficiently computable function $g_A(j, l)$ gives the column index of the lth non-zero element of row j of matrix $\mathbf{A}$.

These Oracles (2) and (3) allow for an efficient simulation of e^{-iAt} via the methods in [18, 30] and are often standard assumptions in the literature.

3.2. Generic quantum gradient descent step

In this section, we describe how to obtain the update of the current candidate $|\mathbf{x}^{(t)}\rangle$ at the tth step of the gradient descent method. To decouple the simulation method with the main result, we use a generic simulation method of $\mathbf{D}^{(t)}$ and specialize in the section thereafter.

Result 1 (Single gradient descent step). Given quantum states $|\mathbf{x}^{(t)}\rangle$ encoding the current solution at time step t to Problem 1 in amplitude encoding to accuracy $0 < \epsilon^{(t)} < 1$. Let $\Lambda_A \geqslant 1$ and the gradient operator corresponding to the state $|\mathbf{x}^{(t)}\rangle$ be given by $\mathbf{D}^{(t)}$ with $\|\mathbf{D}^{(t)}\| \leqslant p\Lambda_A$. Let a quantum algorithm be given such that the controlled operator $e^{-i\,|11\rangle\langle 11|\otimes \mathbf{D}^{(t)}\tau}$ can be simulated to accuracy $0 < \epsilon_D^{(t)} < 1$ using $\mathcal{O}(\mathrm{poly}(p,\, s_A,\, \Lambda_A\tau/\epsilon_D^{(t)}))$ copies of the state $|\mathbf{x}^{(t)}\rangle$ and queries to the oracles for $\mathbf{A}$, and $\tilde{\mathcal{O}}(\mathrm{poly}(p,\, s_A,\, \Lambda_A\tau/\epsilon_D^{(t)})\log N)$ basic quantum gates for each copy. Let a step size be given by $0 < \eta^{(t)} < 1/(2p\Lambda_A)$. Then there exists a quantum algorithm using Oracles (2) and (3) such that a single step of the gradient descent method of step size $\eta^{(t)}$ to prepare an improved normalized quantum state $|\mathbf{x}^{(t+1)}\rangle \propto (|\mathbf{x}^{(t)}\rangle - \eta^{(t)}\mathbf{D}^{(t)}|\mathbf{x}^{(t)}\rangle)$ to accuracy $\epsilon^{(t+1)} = \mathcal{O}(\eta^{(t)}\epsilon_D^{(t)} + \epsilon^{(t)})$ can be performed. This quantum algorithm requires $\mathcal{O}(\mathrm{poly}(p,\, s_A,\, \Lambda_A/\epsilon_D^{(t)}))$ copies of the state $|\mathbf{x}^{(t)}\rangle$ and queries to the oracle of $\mathbf{A}$, and $\tilde{\mathcal{O}}(\mathrm{poly}(p,\, s_A,\, \Lambda_A/\epsilon_D^{(t)})\log N)$ basic quantum gates. The success probability of this algorithm is lower-bounded by $1/16$.

Proof. We drop the superscript $^{(t)}$ for the quantities $\mathbf{D}^{(t)}$, $\eta^{(t)}$, and $\epsilon_D^{(t)}$ in the following. Starting with the current solution, prepare the state

$$|\mathbf{x}^{(t)}\rangle \frac{1}{\sqrt{2}}(|0\rangle_g - i\,|1\rangle_g)|0 \ldots 0\rangle, \tag{8}$$

where the first register contains the state $|\mathbf{x}^{(t)}\rangle$, the second register contains a single ancilla qubit used for the gradient step vector addition (subscript g), and the final register contains ancilla qubits for the gradient matrix multiplication. The operator $\mathbf{D}$ can be multiplied to $|\mathbf{x}^{(t)}\rangle$ conditioned on the first ancilla being in state $|1\rangle_g$ via phase estimation. Let the eigenstates of $\mathbf{D}$ be given by $|u_j(\mathbf{D})\rangle$ and the eigenvalues by $\lambda_j(\mathbf{D})$. For phase estimation, use a register with $\mathcal{O}(\lceil \log(2 + 1/2\epsilon_D)\rceil)$ qubits to resolve eigenvalues to accuracy ϵ_D. Then use $e^{-i\,|11\rangle\langle 11|\otimes \mathbf{D}\tau}$ to apply $\sum_{l=1}^{S_{\mathrm{ph}}}|l\rangle\langle l|(e^{-i\,|1\rangle\langle 1|\otimes \mathbf{D}\Delta t})^l$ for S_{ph} steps. If we choose $S_{\mathrm{ph}}\Delta t = \mathcal{O}(1/\epsilon_D)$ [11], this conditioned application of the matrix exponentials allows to prepare a quantum state

$$\frac{1}{\sqrt{2}}\left(|\mathbf{x}^{(t)}\rangle\,|0\rangle_g|0\ldots 0\rangle\,-\,\mathrm{i}\sum_j \beta_j|u_j(\mathbf{D})\rangle\,|1\rangle_g|\tilde{\lambda}_j(\mathbf{D})\rangle\right). \tag{9}$$

Here, we have used the current state written in the eigenbasis of $\mathbf{D}$, i.e. $|\mathbf{x}^{(t)}\rangle = \sum_j \beta_j|u_j(\mathbf{D})\rangle$, with $\beta_j = \langle u_j(\mathbf{D})|\mathbf{x}^{(t)}\rangle$. Also, $|\tilde{\lambda}_j(\mathbf{D})\rangle$ is the additional register encoding the corresponding eigenvalue $\lambda_j(\mathbf{D})$ in binary representation with accuracy ϵ_D.

Now use another ancilla (subscript d.) For the $|0\rangle_g$ path this ancilla is initialized in the $|1\rangle_d$ state and for the $|1\rangle_g$ path this ancilla is conditionally rotated. Uncomputing the eigenvalue register arrives at the state

$$\begin{aligned}&\frac{1}{\sqrt{2}}|\mathbf{x}^{(t)}\rangle\,|0\rangle_g|1\rangle_d\\&-\frac{\mathrm{i}}{\sqrt{2}}\sum_j \beta_j|u_j(\mathbf{D})\rangle\,|1\rangle_g(\sqrt{1-(\eta\tilde{\lambda}_j(\mathbf{D}))^2}\,|0\rangle_d\,+\,\eta\tilde{\lambda}_j(\mathbf{D})|1\rangle_d).\end{aligned} \tag{10}$$

Here, the rotation is well defined, $\eta\tilde{\lambda}_j(\mathbf{D}) < 1$, as we assumed that $\eta < 1/(2p\Lambda_A)$ and $p\Lambda_A \geqslant \max_j|\lambda_j(\mathbf{D})|$. Now perform a joint measurement of the final two ancillas. The first ancilla is measured in the basis $|\mathrm{yes}\rangle = \frac{1}{\sqrt{2}}(|0\rangle_g\,+\,\mathrm{i}\,|1\rangle_g)$ and $|\mathrm{no}\rangle = \frac{1}{\sqrt{2}}(\mathrm{i}\,|0\rangle_g\,+\,|1\rangle_g)$. The second ancilla is measured in the state $|1\rangle_d$. Thus, if $|\mathrm{yes}\rangle\,|1\rangle_d$ is measured, we arrive at the state

$$\begin{aligned}&\frac{1}{\tilde{C}_D^{(t+1)}}\left(|\mathbf{x}^{(t)}\rangle\,-\,\eta\sum_j \beta_j\tilde{\lambda}_j(\mathbf{D})|u_j(\mathbf{D})\rangle\right)\\&\approx\frac{1}{C_D^{(t+1)}}(|\mathbf{x}^{(t)}\rangle\,-\,\eta\mathbf{D}|\mathbf{x}^{(t)}\rangle)\equiv|\mathbf{x}^{(t+1)}\rangle,\end{aligned} \tag{11}$$

with normalization

$$(C_D^{(t+1)})^2 = 1\,-\,2\eta\langle\mathbf{x}^{(t)}|\mathbf{D}|\mathbf{x}^{(t)}\rangle\,+\,\eta^2\langle\mathbf{x}^{(t)}|\mathbf{D}^2|\mathbf{x}^{(t)}\rangle, \tag{12}$$

and the erroneous version $\tilde{C}_D^{(t+1)}$. See appendix B for a discussion of the approximation in equation (11). Here, $\langle\mathbf{x}^{(t)}|\mathbf{D}|\mathbf{x}^{(t)}\rangle$ refers to the inner product of the current solution $|\mathbf{x}^{(t)}\rangle$ with the gradient at this point and $\Delta_D \coloneqq \sqrt{\langle\mathbf{x}^{(t)}|\mathbf{D}^2|\mathbf{x}^{(t)}\rangle}$ is the gradient vector length. One can express $(C_D^{(t+1)})^2 = 1\,-\,2\eta\Delta_D\cos\varphi_D\,+\,\eta^2\Delta_D^2$, where φ_D is the angle between $|\mathbf{x}^{(t)}\rangle$ and $\mathbf{D}|\mathbf{x}^{(t)}\rangle$. The success probability of the measurement of $|\mathrm{yes}\rangle\,|1\rangle$ is given by

$$P_{\mathrm{yes},1} = \frac{1}{4}(C_D^{(t+1)})^2. \tag{13}$$

For sufficiently bounded step size, this success probability can be lower bounded. The angle $\varphi_D = 0$ achieves the lower bound

$$(C_D^{(t+1)})^2 \geqslant (1\,-\,\eta\Delta_D)^2. \tag{14}$$

With $0 < \eta < 1/(2p\Lambda_A)$ by assumption and $\Delta_D \leqslant p\Lambda_A$ by definition, we have $(C_D^{(t+1)})^2 > 1/4$. Hence it follows that $P_{\mathrm{yes},1} > 1/16$. Thus, the upper bound for the number of repetitions needed is $\mathcal{O}(1/P_{\mathrm{yes},1}) = \mathcal{O}(1)$. □

The quantum state $|\mathbf{x}^{(t+1)}\rangle$ is an approximation to the classical vector $\mathbf{x}^{(t+1)}$, which would be the result of a classical projected gradient descent update departing from a normalized vector $\mathbf{x}^{(t)}$. The error of the updated step is bounded by the error of the previous step $\epsilon^{(t)}$ plus the error of the conditioned matrix multiplication, i.e. $\mathcal{O}(\epsilon^{(t)}\,+\,\eta\epsilon_D^{(t)})$. Given the assumptions stated above, this state can be prepared via phase estimation, an ancilla rotation and measurement. The measurement and post-selection normalizes the quantum state at each step. The resource requirements are obtained by setting $\tau \to \mathcal{O}(1/\epsilon_D^{(t)})$ from phase estimation into the generic simulation method for $\mathbf{D}^{(t)}$. Taking a step size according to the stated assumption, the probability of success and thus the number of repetitions to success are constants.

3.3. Simulating the gradient

In this section, we describe how to quantumly simulate the gradient operator as $\mathrm{e}^{-\mathrm{i}\mathbf{D}t}$ and provide an explicit resource count for the result 1. Here, we omit the index t indicating the current step for readability. We summarize the optimal Hamiltonian simulation result used in this work. Note that $\|\mathbf{A}\|_{\max} \coloneqq \max_{ij}|\mathbf{A}_{ij}|$ and $\|\mathbf{A}\|_{\max} \leqslant \|\mathbf{A}\|$.

Theorem 1 (**Sparse Hamiltonian simulation** [30]). *Given the Oracles (2) and (3) for the* $\mathrm{s_A}$*-sparse matrix* $\mathbf{A}$ *in the vector space of dimension* $\mathrm{N^P}$, $s_A \geqslant 1$. *Assume that the matrix elements are specified to a constant precision that does not affect the overall dependencies of the results. There exists an efficient quantum algorithm to simulate* $\mathrm{e}^{-\mathrm{i}\mathbf{A}\tau}$ *to accuracy* $0 < \epsilon < 1$ *using* $\mathcal{O}(C_A(\tau,\,\epsilon))$ *queries to the oracles and* $\tilde{\mathcal{O}}(C_A(\tau,\,\epsilon)p\log N)$ *single- and two-qubit*

quantum gates, where

$$C_A(\tau,\,\epsilon) \coloneqq \tau s_A \|\mathbf{A}\|_{\max} + \frac{\log(1/\epsilon)}{\log\log(1/\epsilon)}. \tag{15}$$

Using this fundamental result, we discuss the quantum simulation of the gradient operator $\mathbf{D}$, see the definition equation (6). First, note that the scalar coefficients $\mathbf{x}^T\mathbf{A}_i^\alpha\mathbf{x}$ that occur as weighing factors in equations (6) and (7) can be written as the expectation values of operators $\mathbf{A}_i^\alpha$ as $\langle\mathbf{x}|\mathbf{A}_i^\alpha|\mathbf{x}\rangle = \mathrm{tr}\{\mathbf{A}_i^\alpha|\mathbf{x}\rangle\langle\mathbf{x}|\}$. With this relation, we show that the gradient operator can be implemented as a Hamiltonian in a quantum algorithm. The gradient operator $\mathbf{D}$ can be represented as

$$\mathbf{D} = \mathrm{tr}_{1\ldots p-1}\{(|\mathbf{x}\rangle\langle\mathbf{x}|^{\otimes(p-1)} \otimes \mathcal{I})\,\mathbf{M}_D\}, \tag{16}$$

where $|\mathbf{x}\rangle\langle\mathbf{x}|^{\otimes(p-1)} = |\mathbf{x}\rangle\langle\mathbf{x}|\otimes\cdots\otimes|\mathbf{x}\rangle\langle\mathbf{x}|$ is the joint quantum state of $p-1$ copies of $|\mathbf{x}\rangle$. This operator $\mathbf{D}$ acts on another copy of $|\mathbf{x}\rangle$. The auxiliary operator $\mathbf{M}_D$ is independent of $|\mathbf{x}\rangle$ and given by

$$\mathbf{M}_D = \sum_{\alpha=1}^{K}\sum_{j=1}^{p}\left(\bigotimes_{\substack{i=1\\ i\neq j}}^{p}\mathbf{A}_i^\alpha\right)\otimes\mathbf{A}_j^\alpha. \tag{17}$$

More informally stated, the operator $\mathbf{M}_D$ can be represented from the $\mathbf{A}_j^\alpha$, $j = 1\ldots p$, of equation (5) in such a way that for each term in the sum the expectation values of the first $p-1$ subsystems correspond to the desired weighing factor, and the last subsystem remains as the operator acting on another copy of $|\mathbf{x}\rangle$. Note that the order of the factors in the product $\bigotimes_{\substack{i=1\\ i\neq j}}^{p}\mathbf{A}_i^\alpha$ only changes the matrix $\mathbf{M}_D$ but not the operator $\mathbf{D}$. The matrix exponential $\mathrm{e}^{-\mathrm{i}\mathbf{M}_D\tau}$ can be simulated efficiently on a quantum computer with access to the above oracles, as shown in the following result.

Lemma 1. *Let the derivative auxiliary operator $\mathbf{M}_D$ be given by equation (17), with the vector space dimension N^p. Let $\|\mathbf{A}\| \leqslant \Lambda_A$ with $\Lambda_A \geqslant 1$. There exists an efficient quantum algorithm using Oracles (2) and (3) such that $\mathrm{e}^{-\mathrm{i}\mathbf{M}_D\tau}$ can be simulated. Simulating $\mathrm{e}^{-\mathrm{i}\mathbf{M}_D\tau}$ for a time $\tau > 0$ to accuracy $0 < \epsilon < 1$, we require $\mathcal{O}(C_A(p\tau,\,\epsilon))$ queries to the oracles for $\mathbf{A}$ and $\tilde{\mathcal{O}}(p^3\Lambda_A^2\tau^2 C_A(p\tau,\,\epsilon)\log N/\epsilon)$ quantum gates.*

Proof. The matrix $\mathbf{A}$ can be expressed as the tensor decomposition in equation (5). Note that this tensor decomposition is formally used for the proof but not explicitly required for simulating the gradient matrix $\mathbf{M}_D$. The gradient matrix is given by equation (17), where we can simply interchange the summations to obtain

$$\mathbf{M}_D = \sum_{j=1}^{p}\sum_{\alpha=1}^{K}\left(\bigotimes_{\substack{i=1\\ i\neq j}}^{p}\mathbf{A}_i^\alpha\right)\otimes\mathbf{A}_j^\alpha =: \sum_{j=1}^{p} M_j. \tag{18}$$

For the discussion, we assume that $\bigotimes_{\substack{i=1\\ i\neq j}}^{p}\mathbf{A}_i^\alpha$ is ordered such that the pth matrix is swapped with the jth matrix, which results in the same operator $\mathbf{D}$, since the ordering is not important after taking the partial trace. Note that there exist (unitary) permutation matrices Q_j, such that with $M_j = Q_j\mathbf{A}Q_j^\dagger$ we have

$$\mathbf{M}_D = \sum_{j=1}^{p} Q_j\mathbf{A}Q_j^\dagger. \tag{19}$$

Thus, the M_j matrix elements can be obtained from the $\mathbf{A}$ matrix elements. The permutation matrices are specified via

$$\langle i_1\cdots i_j\cdots i_p|M_j|i_1'\cdots i_j'\cdots i_p'\rangle \tag{20}$$

$$= \sum_{\alpha=1}^{K}(\mathbf{A}_1^\alpha)_{i_1 i_1'}\cdots(\mathbf{A}_p^\alpha)_{i_j i_j'}\cdots(\mathbf{A}_j^\alpha)_{i_p i_p'} \tag{21}$$

$$= \sum_{\alpha=1}^{K}(\mathbf{A}_1^\alpha)_{i_1 i_1'}\cdots(\mathbf{A}_j^\alpha)_{i_p i_p'}\cdots(\mathbf{A}_p^\alpha)_{i_j i_j'}, \tag{22}$$

$$= \langle i_1\cdots i_p\cdots i_j|\mathbf{A}|i_1'\cdots i_p'\cdots i_j'\rangle, \tag{23}$$

to be

$$Q_j = \sum_{i_1,\ldots,i_p=1}^{N}|i_1\cdots i_j\cdots i_p\rangle\langle i_1\cdots i_p\cdots i_j|. \tag{24}$$

To simulate a small step $e^{-iM_D\frac{\tau}{m}}$, with $m \geqslant 1$, use the Lie product formula for the sum in equation (19),

$$\left\| e^{-i\,M_D\frac{\tau}{m}} - \prod_{j=1}^{p} Q_j e^{-i\,A\frac{\tau}{m}} Q_j^\dagger \right\| = \mathcal{O}\left(\frac{p^2\Lambda_A^2\tau^2}{m^2}\right), \tag{25}$$

where the sum of p terms leads to an error scaling as p^2, as discussed for example in [31], even if the error of simulating $\mathbf{A}$ is zero. Hence $\mathbf{M}_D$ can be simulated for time $\frac{\tau}{m}$ via a sequence of simulations of $\mathbf{A}$ between the permutation matrices. Each of the p permutation matrices Q_j can be simulated by swapping the two $\log N$-qubit registers corresponding to the respective permutation, i.e. involving $\log N$ swap operations. By equation (25), a single step thus requires $2p \log N$ swap operations.

For multiple steps,

$$\left\| e^{-i\,M_D\tau} - \left(\prod_{j=1}^{p} Q_j e^{-i\,A\frac{\tau}{m}} Q_j^\dagger\right)^m \right\| = \mathcal{O}\left(m \times \frac{p^2\Lambda_A^2\tau^2}{m^2}\right), \tag{26}$$

we take $m = \Theta(p^2\Lambda_A^2\tau^2/\epsilon)$ to achieve final accuracy $\mathcal{O}(\epsilon)$. For the resource count, we effectively have to simulate $e^{-iAp\tau}$ to desired accuracy ϵ, which takes $\mathcal{O}(C_A(p\tau,\ \epsilon))$ queries to the oracles for $\mathbf{A}$ and $\tilde{\mathcal{O}}(C_A(p\tau,\ \epsilon)p \log N)$ quantum gates. In addition, we have $\mathcal{O}(mp \log N) = \mathcal{O}(p^3\Lambda_A^2\tau^2 \log N/\epsilon)$ swap gates. □

We note that improved methods for simulating $\mathbf{M}_D$ can be used if the sparsity function $g_{M_D}(k,\ l)$ can be efficiently computed from the sparsity function of $\mathbf{A}$. In this case, one can expect a similar performance as theorem 1.

The simulation of $\mathbf{M}_D$ is a means to simulate the gradient operator $\mathbf{D}$. The idea is to implement the matrix exponentiation $e^{-iD\Delta t}$ adapting the quantum state exponentiation procedure outlined in [17]. Since $\mathbf{D}$ depends on the current state $|\mathbf{x}\rangle$, we cannot simply use the oracular exponentiation methods of [18]. Instead we exponentiate the sparse matrix $\mathbf{M}_D$ given in equation (17). For a short time Δt, use multiple copies of $|\mathbf{x}\rangle$ and perform a matrix exponentiation of $\mathbf{M}_D$. In the reduced space of the last copy of $|\mathbf{x}\rangle$, we observe the operation

$$\mathrm{tr}_{p-1}\{e^{-iM_D\Delta t} \mid \mathbf{x}\rangle\langle\mathbf{x}|^{\otimes p}\, e^{iM_D\Delta t}\} = e^{-iD\Delta t}|\,\mathbf{x}\rangle\langle\mathbf{x}|\, e^{iD\Delta t} + E, \tag{27}$$

where

$$\mathcal{E} = \mathcal{E}_{M_D} + \mathcal{E}_{\mathrm{samp}}. \tag{28}$$

Here, the error term $\mathcal{E}$ contains contributions from the erroneous simulation of $e^{-i\,M_D\Delta t}$ given by $\mathcal{E}_{M_D}$ and the intrinsic error of the sample-based Hamiltonian simulation given by $\mathcal{E}_{\mathrm{samp}}$. We use the operator norm $\|\cdot\|$ to bound the error terms.

We have assumed perfect states $|\mathbf{x}\rangle$. However, within the gradient descent routine, the states pick up errors due to the imprecise simulation and phase estimation methods. The modification is then that the error terms for a simulation of time Δt become

$$\mathcal{E}' = \mathcal{E}_{M_D} + \mathcal{E}_{\mathrm{samp}} + \mathcal{E}_{|\mathbf{x}\rangle}, \tag{29}$$

in comparison to equation (28). The additional error term is $\mathcal{E}_{|\mathbf{x}\rangle}$.

We discuss a result regarding the simulation with such imperfect states using a particular error model. The error model assumes a fairly strong bound for the exponentiated gradient operator. Concretely, we ideally want to simulate the Hamiltonian equation (16) using a Trotter decomposition into small steps but instead of the exact steps, we can only simulate each step with an error, i.e. we have to use the Hamiltonian $\mathbf{D}_l = \mathrm{tr}_{1\ldots p-1}\{(\bigotimes_{k=1}^{p-1}|\tilde{\mathbf{x}}_{lk}\rangle\langle\tilde{\mathbf{x}}_{lk}| \otimes \mathcal{I})\ \mathbf{M}_D\}$. Here, $|\tilde{\mathbf{x}}_{lk}\rangle$ are the erroneous samples.

Assumption 1 (Gradient errors). Let $\tau > 0$, $m \in \mathbb{Z} > 0$ and $\|\mathbf{D}\|,\ \|\mathbf{D}_l\| \leqslant p\Lambda_A$. Assume the erroneous $\mathbf{D}_l$ are such that

$$\left\| e^{-i\,D\tau} - \prod_{l=1}^{m} e^{-i\,D_l\tau/m} \right\| \leqslant \frac{p^2\Lambda_A^2\tau^2}{m}. \tag{30}$$

Note appendix A for a derivation of a bound that is not sufficient in the sense that the bound is $\mathcal{O}(\tau)$, independent of m. We also note that other errors may be tolerable in practice. For example, in stochastic gradient descent, a gradient is computed from randomly sampling the gradient vector. This leads to surprisingly good learning behavior for neural networks [32, 33]. More generally it has even been shown that the stochasticity in learning problems leads to better generalizability through an indirect regularization of the problem. In many practical situations, sufficiently close to the minimum of a convex optimization problem, a small error in the gradient can nevertheless lead to convergence if the step size is chosen appropriately. A more general and

quantitative analysis of other error models can give further insight into the algorithms behavior and is left for future work.

This assumption leads to a simple accumulation of errors, as the following result shows.

Lemma 2 (Erroneous sample-based Hamiltonian simulation of D). *Given the ideal Hamiltonian* $\mathbf{D}$*, with* $\|\mathbf{D}\| \leqslant p\Lambda_A$*, and the actual Hamiltonians* $\mathbf{D}_l$ *arising from erroneous samples. Here,* $l = 1, \cdots, m$*, where* m *is the number of time steps. With the assumption 1, we can perform Hamiltonian simulation of* $\mathbf{D}$*, i.e. the simulation of* $\mathrm{e}^{-\mathrm{i}\,\mathbf{D}\tau}$ *for a time* $\tau > 0$ *and desired error* $0 < \epsilon < 1$*. The simulation then requires* $m = \mathcal{O}\left(\frac{p^2\Lambda_A^2\tau^2}{\epsilon}\right)$ *time steps and* p *samples at each time step. The algorithm uses at most* $\mathcal{O}\left(\frac{p^2\Lambda_A^2\tau^2}{\epsilon} C_A(p\tau,\ \epsilon)\right)$ *queries to the oracle for* $\mathbf{A}$ *and* $\tilde{\mathcal{O}}\left(\frac{p^3\Lambda_A^2\tau^2}{\epsilon} C_A(p\tau,\ \epsilon)\ \log N\right)$ *quantum gates.*

Proof. We evaluate the difference in desired and actual time evolution. Note that the error of a single time step $\Delta t = \tau/m$ is given by equation (29). As before, we can choose $\|\mathcal{E}_{\mathbf{M}_D}\| = \mathcal{O}(p^2\Lambda_A^2\Delta t^2)$. Regarding the error $\mathcal{E}_{\mathrm{samp}}$, its size is given by $\|\mathcal{E}_{\mathrm{samp}}\| = \mathcal{O}(p^2\Lambda_A^2\Delta t^2)$ similar to [17]. The error for a small time step Δt from these two sources is thus upper bounded by $\|\mathcal{E}\| \leqslant \|\mathcal{E}_{\mathrm{samp}}\| + \|\mathcal{E}_{\mathbf{M}_D}\| = \mathcal{O}(p^2\Lambda_A^2\Delta t^2)$. For m steps this error is $m\|\mathcal{E}\| = \mathcal{O}(p^2\Lambda_A^2\tau^2/m)$. The final step is now to account for the contribution of $\mathcal{E}_{|\mathbf{x}\rangle}$. The error ϵ between desired and actual time evolution for the total time τ is bounded by

$$\epsilon \leqslant \left\| \mathrm{e}^{-\mathrm{i}\,\mathbf{D}\tau} - \prod_{l=1}^{m} \mathrm{e}^{-\mathrm{i}\,\mathbf{D}_l\tau/m} \right\| + m\|\mathcal{E}\| \tag{31}$$

$$= \mathcal{O}\left(\frac{p^2\Lambda_A^2\tau^2}{m}\right). \tag{32}$$

Here, we have used the assumption 1. We obtain for the number of samples

$$m = \mathcal{O}\left(\frac{p^2\Lambda_A^2\tau^2}{\epsilon}\right). \tag{33}$$

□

As discussed, result 1 uses a generic method for simulation of $\mathbf{D}$. A concrete estimate of the resources required to implement the single-step algorithm using the error model assumption 1 is stated as follows.

Result 2. In the setting of result 1, include the assumption 1 for the errors of the derivative operator. The quantum algorithm to perform result 1 then requires $\mathcal{O}\left(\frac{p^3\Lambda_A^2}{(\epsilon_D^{(t)})^3}\right)$ copies of $|\mathbf{x}^{(t)}\rangle$. The required number of queries to the oracle for $\mathbf{A}$ is given by $\mathcal{O}\left(\frac{p^2\Lambda_A^2}{(\epsilon_D^{(t)})^3} C_A(p/\epsilon_D^{(t)},\ \epsilon_D^{(t)})\right)$ and the number of elementary quantum gates is given by $\tilde{\mathcal{O}}\left(\frac{p^3\Lambda_A^2}{(\epsilon_D^{(t)})^3} C_A(p/\epsilon_D^{(t)},\ \epsilon_D^{(t)})\ \log N\right)$.

Proof. The resource requirements for this result are given from lemma 2 for the sample-based Hamiltonian simulation. The extension to the controlled simulation as required by phase estimation is done in an overhead that factors in as a constant into the resource requirements [34]. Replacing $\tau \to \mathcal{O}(1/\epsilon_D^{(t)})$ gives the stated result 2. □

3.4. Multiple steps

We have presented a single step of the quantum version of gradient descent for polynomial optimization. We now estimate the resource requirements for multiple steps.

Result 3 (Multiple gradient steps). Assume the setting of optimization of polynomials given in Problem 1. Let the task be to use the quantum gradient descent method for $T \geqslant 0$ steps to prepare a solution $|\mathbf{x}^{(T)}\rangle$ to final accuracy $0 < \delta < 1$, with step-size schedule $\eta^{(t)} > 0$. Assume a quantum algorithm according to result 1 exists at every step $0 \leqslant t < T$ and $\epsilon^{(0)} = 0$. Assume that the space explored with the algorithm is bounded by a constant, i.e. $T \max_t \eta^{(t)} = \mathcal{O}(1)$. Let $\Lambda_A \geqslant 1$ such that $\|\mathbf{D}^{(t)}\| \leqslant p\Lambda_A$. Then there exists a quantum algorithm for multiple gradient steps that requires

$$\mathcal{O}(\mathrm{poly}(p,\, s_A,\, \Lambda_A/\delta)^T), \tag{34}$$

copies of the initial state $|\mathbf{x}^{(0)}\rangle$. The gate requirement is $\tilde{\mathcal{O}}(\mathrm{poly}(p,\, s_A,\, \Lambda_A/\delta)^T \log N)$ quantum gates. In addition, if the quantum states $|\mathbf{x}^{(t)}\rangle$ at every step satisfy the error model given in assumption 1 and a simulation method for the operator $\mathbf{D}^{(t)}$ is provided according to lemma 2, then there exists a quantum algorithm that requires

$$\mathcal{O}\left(\frac{C_{\mathrm{single}}^T p^{3T} \Lambda_A^{2T}}{\delta^{3T}}\right) \tag{35}$$

copies of the initial state $|\mathbf{x}^{(0)}\rangle$ and $\tilde{\mathcal{O}}\left(\frac{C_{\mathrm{single}}^T p^{3T} \Lambda_A^{2T}}{\delta^{3T}} C_A (p/\delta,\, \delta)^T \log N\right)$ quantum gates. Here, $C_{\mathrm{single}} \geqslant 1$ is a constant specified by the concrete implementation of the single step algorithm.

Proof. When performing multiple steps, $t = 0, \ldots, T$, the step size parameter η is usually decreased as one gets closer to the target, i.e. $\eta \to \eta^{(t)}$. Each step incurs the error $\epsilon^{(t)} = \epsilon^{(t-1)} + \eta^{(t)} \epsilon_D^{(t)}$, i.e. the error from the previous step and the gradient error. Hence the accumulated error is $\epsilon^{(t)} = \epsilon^{(0)} + \sum_{t'=0}^{t-1} \eta^{(t')} \epsilon_D^{(t')}$. At step T this error shall be $\epsilon^{(T)} \leqslant \delta$. To achieve this final error δ, choose the desired error of each gradient multiplication $\epsilon_D^{(t)} = \delta/T\eta^{(t)}$. Using this $\epsilon_D^{(t)}$, we have for the accumulated error $\epsilon^{(t)} = t\delta/T \leqslant \delta$ for $t \leqslant T$. For a single gradient descent step according to result 1, the number of copies required is then, using $\epsilon_D^{(t)} = \delta/T\eta$,

$$\mathcal{O}(\mathrm{poly}(p,\, s_A,\, \Lambda_A/\delta)), \tag{36}$$

where we have used the assumption $T\eta \leqslant T \max_t \eta^{(t)} = \mathcal{O}(1)$. Thus, T iterations of the gradient descent method require,

$$\mathcal{O}(\mathrm{poly}(p,\, s_A,\, \Lambda_A/\delta)^T) \tag{37}$$

copies of the initial state $|\mathbf{x}^{(0)}\rangle$. Each copy requires $\tilde{\mathcal{O}}(\mathrm{poly}(p,\, s_A,\, \Lambda_A/\delta) \log N)$ gates, thus the overall gate requirement is $\tilde{\mathcal{O}}(\mathrm{poly}(p,\, s_A,\, \Lambda_A/\delta)^T \log N)$ quantum gates. Now, take the assumption 1 and the simulation method provided by lemma 2. For a single gradient descent step according to result 2, the number of copies required is then

$$\mathcal{O}\left(\frac{p^3 \Lambda_A^2 (T\eta)^3}{\delta^3}\right) = \mathcal{O}\left(\frac{p^3 \Lambda_A^2}{\delta^3}\right). \tag{38}$$

Let $C_{\mathrm{single}} \geqslant 1$ be an upper bound for the constants omitted by the $\mathcal{O}()$ notation for the number of copies, oracle queries, and gate count of a single step. For T iterations of the gradient descent method, we then need at most

$$\mathcal{O}\left(\frac{C_{\mathrm{single}}^T p^{3T} \Lambda_A^{2T}}{\delta^{3T}}\right) \tag{39}$$

copies of the initial state $|\mathbf{x}^{(0)}\rangle$. The gate requirement is $\tilde{\mathcal{O}}\left(\frac{C_{\mathrm{single}}^T p^{3T} \Lambda_A^{2T}}{\delta^{3T}} C_A (p/\delta,\, \delta)^T \log N\right)$.

□

Thus, for multiple steps a number of copies that is exponentiated by the number of steps is required. Figure 4 illustrates this process. While the upper bound potentially can be improved significantly, it is obvious that any exponential growth of the resources in both space and run time with the number of steps is prohibitive. A possible solution is to look at extensions of the gradient descent method which exhibit faster convergence to the minimum in only a few steps. One such option is Newton's method, which requires inverting the Hessian matrix of $f(\mathbf{x})$ at point $\mathbf{x}$, an operation that can become expensive on a classical computer while being fast on a quantum computer [14].

4. Quantum Newton's method

A quantum algorithm for Newton's method follows the quantum gradient descent scheme, but in addition to the conditional implementation of $\mathbf{D}|\mathbf{x}\rangle = |\nabla f(\mathbf{x})\rangle$, one also applies an operator $\mathbf{H}^{-1}$ to $|\nabla f(\mathbf{x})\rangle$ which represents the inverse Hessian matrix. Our aim (classically and quantumly) here is to invert the well-conditioned subspace of $\mathbf{H}$ and not take into account zero eigenvalues or eigenvalues that are extremely (exponentially) small [11, 14]. Thus in this work, by $\mathbf{H}^{-1}$ we denote the inverse on this well-conditioned subspace. We define the well-conditioned subspace of $\mathbf{H}$ via the effective minimum eigenvalue $\lambda_{\min,\mathrm{eff}}(\mathbf{H})$ such that $\|\mathbf{H}^{-1}\| = 1/\lambda_{\min,\mathrm{eff}}(\mathbf{H}) \leqslant \Lambda_{H^{-1}}$ where $\Lambda_{H^{-1}}$ is a constant.

A Newton step can be performed, assuming a generic simulation method for the Hessian and the gradient operator as before and will be specialized to a concrete estimate in result 5.

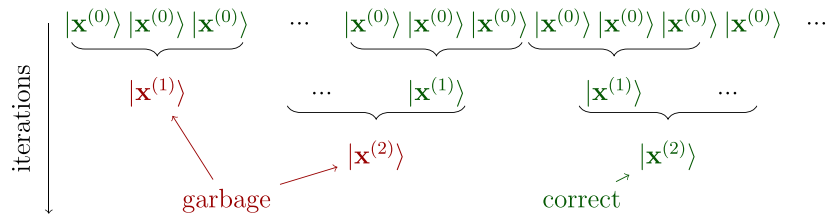

Figure 4. Both the quantum gradient descent and Newton's method 'consume' copies in every step and have only a probability of less than one to produce the correct updated state. This limits their application to local searches with a small amount of iterations and is in principle an underlying feature of iterative quantum routines with probabilistic outcomes.

Result 4 (Single Newton step). Given quantum states $|\mathbf{x}^{(t)}\rangle$ encoding the current solution at time step t to Problem 1 in amplitude encoding to accuracy $0 < \epsilon^{(t)} < 1$. Let the gradient operator corresponding to the state $|\mathbf{x}^{(t)}\rangle$ be given by $\mathbf{D}^{(t)}$ and the Hessian matrix corresponding to the state $|\mathbf{x}^{(t)}\rangle$ be given by $\mathbf{H}^{(t)}$. Let $\Lambda_A \geqslant 1$ and $\Lambda_{H^{-1}} \geqslant 1$ be such that $\|\mathbf{D}^{(t)}\| \leqslant p\Lambda_A$, $\|\mathbf{H}^{(t)}\| \leqslant p^2\Lambda_A$, and $\|(\mathbf{H}^{(t)})^{-1}\| \leqslant \Lambda_{H^{-1}}$. Let a quantum algorithm be given such that the controlled operator $e^{-i\,|11\rangle\langle 11|\otimes \mathbf{D}^{(t)}\tau}$ can be simulated to accuracy $0 < \epsilon^{(t)}_{\mathrm{nwt}} < 1$ using $\mathcal{O}(\mathrm{poly}(p,\, s_A,\, \Lambda_A\tau/\epsilon^{(t)}_{\mathrm{nwt}}))$ copies of the state $|\mathbf{x}^{(t)}\rangle$ and queries to the oracles for $\mathbf{A}$, and $\tilde{\mathcal{O}}(\mathrm{poly}(p,\, s_A,\, \Lambda_A\tau/\epsilon^{(t)}_{\mathrm{nwt}})\log N)$ basic quantum gates for each copy. Furthermore, let a quantum algorithm be given such that the controlled operator $e^{-i\,|11\rangle\langle 11|\otimes \mathbf{H}^{(t)}\tau}$ can be simulated to accuracy $0 < \epsilon^{(t)}_{\mathrm{nwt}} < 1$ using $\mathcal{O}(\mathrm{poly}(p,\, s_A,\, \Lambda_A\tau/\epsilon^{(t)}_{\mathrm{nwt}}))$ copies of the state $|\mathbf{x}^{(t)}\rangle$ and queries to the oracles for $\mathbf{A}$, and $\tilde{\mathcal{O}}(\mathrm{poly}(p,\, s_A,\, \Lambda_A\tau/\epsilon^{(t)}_{\mathrm{nwt}})\log N)$ basic quantum gates for each copy. Let a step size be given by $0 < \sqrt{\eta^{(t)}} < 1/(\Lambda\sqrt{2})$, where $\Lambda := \max\{p\Lambda_A,\, \Lambda_{H^{-1}}\}$. Then there exists a quantum algorithm using Oracles (2) and (3) such that a single step of the Newton's method of step size $\eta^{(t)}$ to prepare an improved normalized quantum state $|\mathbf{x}^{(t+1)}\rangle \propto (|\mathbf{x}^{(t)}\rangle - \eta^{(t)}(\mathbf{H}^{(t)})^{-1}\mathbf{D}^{(t)}|\mathbf{x}^{(t)}\rangle)$ to accuracy $\epsilon^{(t+1)} = \mathcal{O}(\eta^{(t)}\epsilon^{(t)}_{\mathrm{nwt}} + \epsilon^{(t)})$ can be performed. This quantum algorithm requires $\mathcal{O}(\mathrm{poly}(p,\, s_A,\, \Lambda/\epsilon^{(t)}_{\mathrm{nwt}}))$ copies of the state $|\mathbf{x}^{(t)}\rangle$ and queries to the oracle of $\mathbf{A}$, and $\tilde{\mathcal{O}}(\mathrm{poly}(p,\, s_A,\, \Lambda/\epsilon^{(t)}_{\mathrm{nwt}})\log N)$ basic quantum gates. The success probability of this algorithm is lower-bounded by 1/16.

Proof. The quantum Newton step proceeds as follows, similar to the gradient descent step. Starting with the current solution, prepare the state

$$|\mathbf{x}^{(t)}\rangle \frac{1}{\sqrt{2}}(|0\rangle_g - \mathrm{i}\,|1\rangle_g)|0\ldots 0\rangle|0\ldots 0\rangle, \tag{40}$$

where the final registers contain ancilla qubits for the gradient matrix multiplication and the Hessian matrix inversion. With the eigenstates $|u_j(\mathbf{D})\rangle$ and the eigenvalues $\lambda_j(\mathbf{D})$ of $\mathbf{D}$, the gradient operator phase estimation obtains

$$\frac{1}{\sqrt{2}}\left(|\mathbf{x}^{(t)}\rangle|0\rangle_g|0\ldots 0\rangle - \mathrm{i}\sum_j \beta_j|u_j(\mathbf{D})\rangle|1\rangle_g|\tilde{\lambda}_j(\mathbf{D})\rangle|0\ldots 0\rangle\right), \tag{41}$$

where $\beta_j = \langle u_j(\mathbf{D})|\mathbf{x}^{(t)}\rangle$. Here, $\tilde{\lambda}_j(\mathbf{D})$ is an approximation to $\lambda_j(\mathbf{D})$ with accuracy ϵ_{nwt}. With the eigenstates of $\mathbf{H}$ given by $|u_j(\mathbf{H})\rangle$ and the eigenvalues by $\lambda_j(\mathbf{H})$, the Hessian operator phase estimation conditioned on the ancilla being $|1\rangle_g$ obtains

$$\frac{1}{\sqrt{2}}\left(|\mathbf{x}^{(t)}\rangle|0\rangle_g|0\ldots 0\rangle - \mathrm{i}\sum_{j,j'}\beta_j\beta_{jj'}|u_{j'}(\mathbf{H})\rangle|1\rangle_g|\tilde{\lambda}_j(\mathbf{D})\rangle|\tilde{\lambda}_{j'}(\mathbf{H})\rangle\right). \tag{42}$$

Here, $\beta_{jj'} = \langle u_{j'}(\mathbf{H})|u_j(\mathbf{D})\rangle$ and $\tilde{\lambda}_j(\mathbf{H})$ is an approximation to $\lambda_j(\mathbf{H})$ with accuracy ϵ_{nwt}. Perform the conditional rotation of an ancilla (subscript d) for the derivative operator

$$|1\rangle_g|\tilde{\lambda}_j(\mathbf{D})\rangle|0\rangle_d \to |1\rangle_g|\tilde{\lambda}_j(\mathbf{D})\rangle(\sqrt{1-(\sqrt{\eta}\,\tilde{\lambda}_j(\mathbf{D}))^2}\,|0\rangle_d + \sqrt{\eta}\,\tilde{\lambda}_j(\mathbf{D})|1\rangle_d). \tag{43}$$

Here, $\sqrt{\eta}\,\tilde{\lambda}_j(\mathbf{D}) < 1$ for all j, as we have assumed that $0 < \sqrt{\eta} < 1/(\Lambda\sqrt{2})$, where $\Lambda := \max\{p\Lambda_A,\, \Lambda_{H^{-1}}\}$. In addition, perform a conditional rotation of another ancilla (subscript h) for the eigenvalues in the well-conditioned subspace of $\mathbf{H}$ [14]

$$|1\rangle_g|\tilde{\lambda}_{j'}(\mathbf{H})\rangle|0\rangle_h \to |1\rangle_g|\tilde{\lambda}_{j'}(\mathbf{H})\rangle\left(\sqrt{1-\left(\frac{\sqrt{\eta}}{\tilde{\lambda}_{j'}(\mathbf{H})}\right)^2}\,|0\rangle_h + \frac{\sqrt{\eta}}{\tilde{\lambda}_{j'}(\mathbf{H})}|1\rangle_h\right). \tag{44}$$

Here, $\frac{\sqrt{\eta}}{\tilde{\lambda}_{j'}(\mathbf{H})} < 1$ for all j' in the well-conditioned subspace, as we have assumed that $0 < \sqrt{\eta} < 1/(\Lambda\sqrt{2}) \leqslant \lambda_{\min,\mathrm{eff}}(\mathbf{H})$. For the $|0\rangle_g$ path, we set the ancillas to $|1\rangle_d|1\rangle_h$ The next step uncomputes the eigenvalue registers by

running the phase estimations in reverse. With $|\text{yes}\rangle = \frac{1}{\sqrt{2}}(|0\rangle_g + \mathrm{i}\,|1\rangle_g)$, a combined measurement of the ancillas in the state $|\text{yes}\rangle\,|1\rangle_d|1\rangle_h$ arrives at

$$\frac{1}{\tilde{C}_H^{(t+1)}}\left(|\mathbf{x}^{(t)}\rangle - \eta\sum_{jj'}\beta_j\beta_{jj'}\frac{\tilde{\lambda}_j(\mathbf{D})}{\tilde{\lambda}_{j'}(\mathbf{H})}|u_{j'}(\mathbf{H})\rangle\right) \tag{45}$$

$$\approx \frac{1}{C_H^{(t+1)}}(|\mathbf{x}\rangle - \eta\mathbf{H}^{-1}\mathbf{D}|\mathbf{x}\rangle) \equiv |\mathbf{x}^{(t+1)}\rangle. \tag{46}$$

Here, the normalization factor is

$$(C_H^{(t+1)})^2 = 1 - 2\eta\langle\mathbf{x}^{(t)}|\mathbf{H}^{-1}\mathbf{D}|\mathbf{x}^{(t)}\rangle + \eta^2\langle\mathbf{x}^{(t)}|\mathbf{D}\mathbf{H}^{-2}\mathbf{D}|\mathbf{x}^{(t)}\rangle, \tag{47}$$

and $\tilde{C}_H^{(t+1)}$ is the erroneous version. In contrast to the gradient descent step, see equation (11), the Newton direction is used to update the previous state. The probability of success of the $|\text{yes}\rangle\,|1\rangle_d|1\rangle_h$ measurement is given by

$$P_{\text{yes},1,1}^{\text{nwt}} = \frac{1}{4}(C_H^{(t+1)})^2. \tag{48}$$

With a similar argument as in result 1 we can bound the number of repetitions needed. One can express $(C_H^{(t+1)})^2 = 1 - 2\eta\Delta_H\cos\varphi_H + \eta^2\Delta_H^2$, where $\Delta_H := \sqrt{\langle\mathbf{x}^{(t)}|\mathbf{D}\mathbf{H}^{-2}\mathbf{D}|\mathbf{x}^{(t)}\rangle}$ and φ_H is the angle between $|\mathbf{x}^{(t)}\rangle$ and $\mathbf{H}^{-1}\mathbf{D}|\mathbf{x}^{(t)}\rangle$. The angle $\varphi_H = 0$ achieves the lower bound

$$(C_H^{(t+1)})^2 \geqslant (1 - \eta\Delta_H)^2. \tag{49}$$

With $0 < \sqrt{\eta} < 1/(\Lambda\sqrt{2})$ by assumption and $\Delta_H \leqslant p\Lambda_A\Lambda_{H^{-1}} \leqslant \Lambda^2$ by definition, we have $(C_H^{(t+1)})^2 \geqslant 1/4$ and $P_{\text{yes},1,1}^{\text{nwt}} > 1/16$. Thus, the upper bound for the number of repetitions needed is $\mathcal{O}(1/P_{\text{yes},1,1}^{\text{nwt}}) = \mathcal{O}(1)$. □

Analogous to before, we can concretely specify the resources via sample-based simulation methods of the gradient operator and the Hessian. The Hessian is given from equation (7) by

$$\mathbf{H} = \mathbf{H}_1 + \mathbf{D}, \tag{50}$$

with $\mathbf{H}_1$ and $\mathbf{D}$ as above. The norm of the Hessian $\mathbf{H}$ shall be bounded as $\|\mathbf{H}\| \leqslant p^2\Lambda_A$, which also bounds the norm of $\mathbf{H}_1$. To obtain the eigenvalues of $\mathbf{H}$ via phase estimation, we exponentiate $\mathbf{H}$ via exponentiating the matrices $\mathbf{H}_1$ and $\mathbf{D}$ sequentially using the standard Lie product formula [31]

$$\mathrm{e}^{\mathrm{i}\,\mathbf{H}\Delta t} = \mathrm{e}^{\mathrm{i}\,\mathbf{H}_1\Delta t}\mathrm{e}^{\mathrm{i}\,\mathbf{D}\Delta t} + \mathcal{O}(p^4\Lambda_A^2\Delta t^2). \tag{51}$$

To implement the individual exponentiations themselves we use a similar trick as before. We associate a simulatable auxiliary operator $\mathbf{M}_{H_1}$ with $\mathbf{H}_1$ and, as before, the operator $\mathbf{M}_D$ with $\mathbf{D}$. For the part $\mathbf{H}_1$, the corresponding operator $\mathbf{M}_{H_1}$ is given by

$$\mathbf{M}_{H_1} = 2\sum_{\alpha=1}^{K}\sum_{j\neq k}^{p}\left(\bigotimes_{i\neq j,k}^{p}\mathbf{A}_i^\alpha\right)\otimes[(\mathcal{I}\otimes\mathbf{A}_k^\alpha)S(\mathcal{I}\otimes\mathbf{A}_j^\alpha)]. \tag{52}$$

Here, S is the swap matrix between the last two registers. Let σ be an arbitrary state on which the matrix exponential $\mathrm{e}^{\mathrm{i}\,\mathbf{H}_1\Delta t}$ shall be applied on and $|\mathbf{x}\rangle$ be the current state. The relationship between the operator $\mathbf{H}_1$ and $\mathbf{M}_{H_1}$ is given by

$$\mathbf{H}_1\sigma = \mathrm{tr}_{1\ldots p-1}\{\mathbf{M}_{H_1}(|\mathbf{x}\rangle\langle\mathbf{x}|^{\otimes p-1}\otimes\sigma)\}, \tag{53}$$

similar to equation (16). The matrix $\mathbf{M}_{H_1}$ can be simulated via simulations of $\mathbf{A}$.

Lemma 3. *Let the Hessian auxiliary operator part $\mathbf{M}_{H_1}$ be given by equation (52), with the vector space dimension N^{p}. Let $\|\mathbf{A}\| \leqslant \Lambda_A$. There exists an efficient quantum algorithm using Oracles (2) and (3) such that $\mathrm{e}^{-\mathrm{i}\,\mathbf{M}_{H_1}t}$ can be simulated. For simulating $\mathrm{e}^{-\mathrm{i}\mathbf{M}_{H_1}t}$ for a time τ to accuracy ϵ, we require $\mathcal{O}(C_A(p^2\tau,\,\epsilon))$ queries to the oracles of $\mathbf{A}$ and $\mathcal{O}(p^6\Lambda_A^2\tau^2\;C_A(p^2\tau,\,\epsilon)\log N/\epsilon)$ quantum gates.*

Proof. Note that the operator can be written as

$$\mathbf{M}_{H_1} = 2\sum_{j\neq k}^{p}H_{jk}, \tag{54}$$

with

$$H_{jk} = \sum_{\alpha=1}^{K}\left(\bigotimes_{i\neq j,k}^{p}\mathbf{A}_i^\alpha\right)\otimes[(\mathcal{I}\otimes\mathbf{A}_k^\alpha)S(\mathcal{I}\otimes\mathbf{A}_j^\alpha)]. \tag{55}$$

Note further that we can relate each H_{jk} to the matrix $\mathbf{A}$, (for $j < k$)

$$\langle i_1 \cdots i_j \cdots i_k \cdots i_{p-1} i_p | H_{jk} | i'_1 \cdots i'_j \cdots i'_k \cdots i'_{p-1} i'_p \rangle = \tag{56}$$

$$= \sum_{\alpha=1}^{K} (\mathbf{A}_1^{\alpha})_{i_1 i'_1} \cdots (\mathbf{A}_p^{\alpha})_{i_j i'_j} \cdots (\mathbf{A}_{p-1}^{\alpha})_{i_k i'_k} \cdots (\mathbf{A}_k^{\alpha})_{i_p i'_{p-1}} (\mathbf{A}_j^{\alpha})_{i_{p-1} i'_p} \tag{57}$$

$$= \langle i_1 \cdots i_{p-1} \cdots i_p \cdots i_k i_j | \mathbf{A} | i'_1 \cdots i_p \cdots i'_{p-1} \cdots i'_k i'_j \rangle. \tag{58}$$

The permutation between the $p-1$ and p part is owed to the additional swap matrix in equation (55). Each required permutation to relate $\mathbf{M}_{H_l}$ to $\mathbf{A}$ can be performed with $2 \log N$ swap operators.

We use the Lie product formula as in lemma 1, to simulate a small time step $\mathrm{e}^{-\mathrm{i}\,\mathbf{M}_{H_l}\frac{\tau}{m}}$ efficiently with error $\mathcal{O}(p^4 \Lambda_A^2 \tau^2 / m^2)$ using p^2 small-time simulations of $\mathbf{A}$. We also have $2p^2 \log N$ swap operations for a small time step.

For multiple steps, choose $m = \Theta(p^4 \Lambda_A^2 \tau^2 / \epsilon)$ to achieve a given error ϵ. For the resource count, we effectively have to simulate $\mathrm{e}^{-\mathrm{i}\,\mathbf{A} p^2 \tau}$ to desired accuracy ϵ, which takes $\mathcal{O}(C_A(p^2\tau,\ \epsilon))$ queries to the oracles for $\mathbf{A}$ and $\tilde{\mathcal{O}}(C_A(p^2\tau,\ \epsilon) p \log N)$ quantum gates. In addition, we have $\mathcal{O}(m p^2 \log N) = \mathcal{O}(p^6 \Lambda_A^2 \tau^2 \log N / \epsilon)$ swap gates. □

We can use multiple copies of the current state $|\mathbf{x}\rangle$ to perform

$$\begin{aligned} &\mathrm{tr}_{1\ldots p-1}\{\mathrm{e}^{-\mathrm{i}\,\mathbf{M}_{H_l}\Delta t}\ (|\mathbf{x}\rangle\langle\mathbf{x}| \otimes \cdots \otimes |\mathbf{x}\rangle\langle\mathbf{x}|)\ \otimes\ \sigma\ \mathrm{e}^{\mathrm{i}\,\mathbf{M}_{H_l}\Delta t}\} \\ &= \mathcal{I} - \mathrm{i}\,\Delta t\,[\mathbf{H}_l,\ \sigma] + \mathcal{O}(\Delta t^2) \approx \mathrm{e}^{-\mathrm{i}\,\mathbf{H}_l \Delta t}\ \sigma\ \mathrm{e}^{\mathrm{i}\,\mathbf{H}_l \Delta t}. \end{aligned} \tag{59}$$

The error for a small time step arises from the sample-based simulation method, the Lie product formula, and from errors from the current solution $|\mathbf{x}\rangle$.

Assumption 2 (Hessian errors). Let $\tau > 0$, $m \in \mathbb{Z} > 0$ and $\|\mathbf{H}_l\|,\ \|(\mathbf{H}_l)_l\| \leqslant p^2 \Lambda_A$. Assume the erroneous $(\mathbf{H}_l)_l$ are such that

$$\left\| \mathrm{e}^{-\mathrm{i}\,\mathbf{H}_l \tau} - \prod_{l=1}^{m} \mathrm{e}^{-\mathrm{i}\,(\mathbf{H}_l)_l \tau / m} \right\| \leqslant \frac{p^4 \Lambda_A^2 \tau^2}{m}. \tag{60}$$

Lemma 4 (Erroneous sample-based Hamiltonian simulation of $\mathbf{H}_l$). *Given the desired Hamiltonian $\mathbf{H}_l$, with $\|\mathbf{H}_l\| \leqslant \|\mathbf{H}\| \leqslant p^2 \Lambda_A$, and the actual Hamiltonians $(\mathbf{H}_l)_l$ arising from erroneous samples. With the assumption 2, the simulation of $\mathrm{e}^{-\mathrm{i}\,\mathbf{H}_l \tau}$ for a time $\tau > 0$ and desired error $0 < \epsilon < 1$ can be performed, with $m = \mathcal{O}\left(\frac{p^4 \Lambda_A^2 \tau^2}{\epsilon}\right)$ time steps. The number of samples needed at each time step is p. The algorithm uses $\mathcal{O}\left(\frac{p^4 \Lambda_A^2 \tau^2}{\epsilon} C_A(p^2\tau,\ \epsilon)\right)$ queries to the oracle for $\mathbf{A}$ and $\tilde{\mathcal{O}}\left(\frac{p^6 \Lambda_A^2 \tau^2}{\epsilon} C_A(p^2\tau,\ \epsilon)\ \log N\right)$ quantum gates.*

Proof. The proof is analogous to lemma 2, but using lemma 3 for the simulation of $M_{\mathbf{H}_l}$. □

Result 5. In the setting of result 4, include assumptions 1 and 2 for the errors of the operators $\mathbf{D}^{(t)}$ and $\mathbf{H}^{(t)}$. The quantum algorithm to perform result 4 then requires $\mathcal{O}\left(\frac{p^5 \Lambda_A^2}{\epsilon_{\mathrm{nwt}}^3}\right)$ copies of $|\mathbf{x}^{(t)}\rangle$. The required number of queries to the oracle for $\mathbf{A}$ is given by $\mathcal{O}\left(\frac{p^4 \Lambda_A^2}{\epsilon_{\mathrm{nwt}}^3} C_A(p^2/\epsilon_{\mathrm{nwt}},\ \epsilon_{\mathrm{nwt}})\right)$ and the number of elementary operations is given by $\tilde{\mathcal{O}}\left(\frac{p^6 \Lambda_A^2}{\epsilon_{\mathrm{nwt}}^3} C_A(p^2/\epsilon_{\mathrm{nwt}},\ \epsilon_{\mathrm{nwt}})\ \log N\right)$.

For the Newton's method the performance is determined from computing the gradient, inverting the Hessian, and subsequent vector addition as before. If the step-size is chosen appropriately, the post-selection for the gradient, Hessian, and taking the Newton step succeeds with constant probability. The phase estimation for the gradient to accuracy ϵ_{nwt} requires $\mathcal{O}\left(\frac{p^3 \Lambda_A^2}{\epsilon_{\mathrm{nwt}}^3}\right)$ copies. The phase estimation for the Hessian to accuracy ϵ_{nwt} requires $\mathcal{O}\left(\frac{p^5 \Lambda_A^2}{\epsilon_{\mathrm{nwt}}^3}\right)$ copies. The two phase estimation requirements contribute additively and thus the number of copies for a single Newton step is

$$\mathcal{O}\left(\frac{p^5 \Lambda_A^2}{\epsilon_{\mathrm{nwt}}^3}\right) \tag{61}$$

copies. The required number queries to the oracle for $\mathbf{A}$ is given by $\mathcal{O}\left(\frac{p^4 \Lambda_A^2}{\epsilon_{\mathrm{nwt}}^3} C_A(p^2/\epsilon_{\mathrm{nwt}},\ \epsilon_{\mathrm{nwt}})\right)$ and the number of elementary operations is given by $\tilde{\mathcal{O}}\left(\frac{p^6 \Lambda_A^2}{\epsilon_{\mathrm{nwt}}^3} C_A(p^2/\epsilon_{\mathrm{nwt}},\ \epsilon_{\mathrm{nwt}})\ \log N\right)$, adding the requirements for gradient and Hessian. The improved solution is prepared to accuracy $\mathcal{O}(\eta \epsilon_{\mathrm{nwt}} + \epsilon^{(t)})$, as the current solution is given to accuracy $\epsilon^{(t)}$.

Similar to the discussion on multiple steps for gradient descent above, for T iterations of Newton's method to final accuracy δ, we need at most

$$\mathcal{O}\left(\frac{C_{\text{single}}^{T} p^{5T} \Lambda_A^{2T}}{\epsilon_{\text{nwt}}^{3T}}\right) \tag{62}$$

copies of the initial state $|\mathbf{x}^{(0)}\rangle$. The gate complexity is $\tilde{\mathcal{O}}\left(\frac{C_{\text{single}}^{T} p^{6T} \Lambda_A^{2T}}{\epsilon_{\text{nwt}}^{3T}} C_A^T (p^2/\epsilon_{\text{nwt}},\, \epsilon_{\text{nwt}})\, \log N\right)$, logarithmic in the dimension N. Here, C_{single}^{T} is again a constant upper bounding the constants omitted by the $\mathcal{O}()$ notation for the number of copies, oracle queries, and gate count of the single Newton step.

This means that the required number of copies of the initial state depends exponentially on the number of steps T. However, recall that in Newton's method in the vicinity of an optimal solution $\mathbf{x}^*$ the accuracy $\Delta := |\mathbf{x}^* - \mathbf{x}^{(T)}|$ of the estimate often improves quadratically with the number of iterations, $\Delta \propto \mathcal{O}(\exp(-T^2))$. This convergence is for example discussed for unconstrained convex problems in [3]. Theorem 3.5 in [4] shows that if the function is twice differentiable, Lipschitz continuous in the neighborhood of a solution, and the Hessian positive definite at the solution then the convergence of Newton's method is quadratic. See also second-order sufficient conditions in theorem 2.4 therein. For projected methods, the convergence properties often translate under similar conditions. For example for optimizing inequality constraints via Newton's method one obtains a quadratic convergence [35]. For proximal Newton methods, which are generalized versions of projected Newton methods, one obtains quadratic convergence under the assumption of strong convexity around the optimum and Lipshitz continuity of the Hessian [36]. Thus in some cases, even though the quantum algorithm requires a number of initial copies that grows exponentially in the number of iterations required, the accuracy of the approximate solution yielded by the algorithm can improve even faster in the number of iterations.

5. Inhomogeneous polynomials

Our methods can be extended to polynomials that are of odd degree and also inhomogeneous. An example of a homogeneous polynomial of odd degree is $x_1^5 x_2^2 + x_1^3 x_2^4$ and an example of an inhomogeneous polynomial is $x_1^5 x_2^2 + x_1^2 x_2^2$. A detailed discussion of such polynomials will be left for future work. We provide a discussion for a subset of polynomials we can solve in a similar fashion to the homogeneous, even polynomials in Problem 1. The problem can be posed as follows.

Problem 2. Let $f(\mathbf{x})$ be a polynomial for which

$$f(\mathbf{x}) = f_{\text{hom,even}}(\mathbf{x}) + f_{\text{inhom}}(\mathbf{x}), \tag{63}$$

where $f_{\text{hom,even}}(\mathbf{x})$ is as in Problem 1. Let the inhomogeneous part of the objective function be given by

$$f_{\text{inhom}}(\mathbf{x}) = \sum_{j=1}^{p-1} (\mathbf{c}_j^T \mathbf{x}) \prod_{i=1}^{j-1} (\mathbf{x}^T B_{ij} \mathbf{x}), \tag{64}$$

where the vector $\mathbf{c}_j$ and the symmetric matrices B_{ij} define the polynomial. Given an initial guess $\mathbf{x}_0$, attempt to solve the problem

$$\min_{\mathbf{x}} f(\mathbf{x}),$$

subject to the constraint $\mathbf{x}^T\mathbf{x} = 1$ by finding a local minimum $\mathbf{x}^*$. Let the assumptions of Problem 1 also apply to the problem defined by $f(\mathbf{x})$.

The term $f_{\text{inhom}}(\mathbf{x})$ allows us to represent a class of monomials of uneven degree $\leqslant 2p - 1$. This class is essentially a sum of homogeneous even polynomials $\prod_{i=1}^{j-1} (\mathbf{x}^T B_{ij} \mathbf{x})$, with the term $\mathbf{c}_j^T \mathbf{x}$ adding inhomogeneities. Thus, with $f_{\text{hom,even}}(\mathbf{x})$ from above, we can represent combinations of homogeneous even polynomials with additional inhomogeneous terms In practice, the efficient sparse Hamiltonian simulation methods impose restrictions on the number of inhomogeneous terms that can be efficiently optimized. Sparsity of the matrices B_{ij} implies a relatively small number of inhomogeneous monomials.

The optimization of functions containing terms of the form $f_{\text{inhom}}(\mathbf{x})$ can be performed via additional simulation terms and vector additions. In order to perform the inhomogeneous updates we require the following ingredients.

- We analytically compute the respective gradient and Hessian of the function $f_{\text{inhom}}(\mathbf{x})$ similar to the homogeneous part.

- We simulate the time evolution under the respective gradient and Hessian of the function $f_{\text{inhom}}(\mathbf{x})$, using similar quantum state exponentiation methods as described in the homogeneous setting. We have to use additional subroutines to simulate terms that contain inner products of the form $\langle \mathbf{x}|\mathbf{c}_j\rangle$, and outer products such as $|\mathbf{x}\rangle\langle\mathbf{c}_j| + |\mathbf{c}_j\rangle\langle\mathbf{x}|$.
- We perform additional vector additions related to the states $\mathbf{c}_j$. For example the function $f_{\text{hom,even}}(\mathbf{x}) + \mathbf{c}^T\mathbf{x}$ requires one additional vector addition. Before adding the vectors we have to conditionally apply the gradient and Hessian operators to the $\mathbf{x}$ and $\mathbf{c}_j$ respectively. This requires in each iteration a number of copies of the $\mathbf{c}_j$ and $\mathbf{x}$. As we require in every state another copy of the states this adds another tree of states to the resource requirements, since we need to build these in parallel to the main algorithm.
- We perform operations similar to the homogeneous step matrix-vector multiplications and matrix inversions via a conditional rotations on the eigenvalue registers and postselection. This will result in different success probabilities for the gradient and the Hessian operator than presented above.
- Finally we perform a measurement in the yes/no-basis as before and perform the vector addition of the current solution and the step update.

All of these steps will add a computational overhead due to additional matrix multiplications and vector additions. This computational overhead will be discussed in a future work. Similarly to the homogeneous case the number of computational steps scales exponentially with the number of iterations T and logarithmically in the dimension of the solution vector.

6. Discussion and conclusion

The present work has considered iterative polynomial optimization in a quantum computing framework. We have developed quantum algorithms for the optimization of a class of polynomials under spherical constraints, for which we can find expressions for the gradient and the Hessian matrix. The class of polynomials we can optimize is constrained by sparsity conditions of Hamiltonian simulation methods used here. Beyond polynomials, one can envision a setting where copies of quantum states representing the current solution are consumed for evaluating the first derivative of the objective function. If we can implement the operation

$$|\mathbf{x}\rangle\otimes\cdots\otimes|\mathbf{x}\rangle\,|0\rangle \mapsto |\mathbf{x}\rangle\otimes\cdots\otimes|\mathbf{x}\rangle\,|\nabla f(\mathbf{x})\rangle, \tag{65}$$

we can use the same basic gradient descent steps as discussed in this work with a similar performance as presented here for polynomial optimization.

In reference [37], Childs *et al* presented an exponential improvement of the matrix inversion error dependence, $1/\epsilon \to \log 1/\epsilon$ by using approximation polynomials instead of phase estimation. In addition, variable-time amplitude amplification can lead to further quadratic speedups [37–39]. Our optimization algorithms scale exponentially in the number of steps performed. We envision a few scenarios where the algorithm can nevertheless be useful. One scenario is when the size of the vectors dominates the dependence on other parameters such as condition number and error raised to the power of the number of steps. In this case, classical computation is prohibitively expensive, while our quantum method scales logarithmically in the size of the vectors and could allow optimization. Often, a constant number of steps can yield a significant improvement on the initial guess, even without finding a local minimum, whereas the problem would be intractable on a classical computer. For (locally) convex problems, the number of iterations of Newton's method required to find a highly accurate solution often is only weakly dependent on the dimension of the system [3] and the minimum can be found in around 5–20 steps. Yet the standard Newton's method is also well known to fail in high-dimensional spaces due to the prevalence of saddle-points, see for example [2]. While this issue is classically not trivial to solve, our quantum method allows for an easy extension to the saddle-free Newton's method [40]. This extension is done by simply replacing the eigenvalues of the Hessian with their absolute values. Thereby a saddle-point behaves like an ordinary minimum and the algorithm takes steps in the correct direction. Therefore, our quantum algorithm is applicable to a wider range of problems with only slight adaptions.

In summary, the optimization algorithms presented here yield a performance $\mathcal{O}(\text{polylog}(N))$ in the dimension N of the vector space over which the optimization is performed, as long as the number of iterations required is small. When searching for a small number of solutions in a featureless landscape, a large number of iterations is required, and the algorithms scale polynomially in N: in particular, the algorithms cannot do better than Grover search, which finds a solution in an essentially 'gradient-free' landscape in time $O(\sqrt{N})$. Further research may find a possible application of quantum gradient algorithms in machine learning for the training of deep neural networks [32, 33], which exhibit a large number of parameters. Gradient descent methods indeed lead to good solutions after a few iterations, and quantum gradient descent algorithms may provide exponential improvements over their classical counterparts.

Acknowledgments

We thank George Siopsis and Simon Benjamin for valuable discussions. We also thank two anonymous referees for their detailed reading of the manuscript and valuable comments. PR acknowledges support from Singapore's Ministry of Education and National Research Foundation. MS and FP acknowledge support by the South African Research Chair Initiative of the Department of Science and Technology and National Research Foundation. SL was supported by ARO.

Appendix A. Assumption on erroneous Hamiltonians

We argue that a simple analysis finds only a relatively bad estimate for the error as

$$\left\| \mathrm{e}^{-\mathrm{i}\,\mathrm{D}\tau} - \prod_{l=1}^{m} \mathrm{e}^{-\mathrm{i}\,(\mathrm{D}+\tilde{\mathrm{D}}_l)\tau/m} \right\| \leqslant \sum_{l=1}^{m} \|\tilde{\mathrm{D}}_l\| \, \frac{\tau}{m}. \tag{A.1}$$

We show this via the following analysis. Let in the following

$$f(s) \coloneqq \prod_{l=1}^{m} \mathrm{e}^{-\mathrm{i}\,(\mathrm{D}+s\tilde{\mathrm{D}}_l)\tau/m} \tag{A.2}$$

be such that $f(0) = \mathrm{e}^{-\mathrm{i}\,\mathrm{D}\tau}$ and $f(1) = \prod_{l=1}^{m} \mathrm{e}^{-\mathrm{i}\,(\mathrm{D}+\tilde{\mathrm{D}}_l)\tau/m}$, so that we can write the term as

$$\|f(0) - f(1)\| = \left\| \int_0^1 \mathrm{d}s \frac{\mathrm{d}f(s)}{\mathrm{d}s} \right\|. \tag{A.3}$$

Then, taking the derivative of $f(s)$ yields

$$\begin{aligned} \frac{\mathrm{d}f(s)}{\mathrm{d}s} = \sum_{k=1}^{m} \left(\prod_{l=1}^{k-1} \mathrm{e}^{-\mathrm{i}\,(\mathrm{D}+s\tilde{\mathrm{D}}_l)\tau/m} \right) \\ \times \left(\int_0^1 \mathrm{d}\hat{s} \mathrm{e}^{-\mathrm{i}\,(\mathrm{D}+s\tilde{\mathrm{D}}_k)(\tau/m)\hat{s}} \left(-\mathrm{i}\,\tilde{\mathrm{D}}_k \frac{\tau}{m} \right) \mathrm{e}^{-\mathrm{i}\,(\mathrm{D}+s\tilde{\mathrm{D}}_k)(\tau/m)(1-\hat{s})} \right) \times \left(\prod_{l=k+1}^{m} \mathrm{e}^{-\mathrm{i}\,(\mathrm{D}+s\tilde{\mathrm{D}}_l)\tau/m} \right), \end{aligned} \tag{A.4}$$

where we used Duhamel's formula (see e.g. [41]) in order to evaluate the operator derivative. We can then bound the term by

$$\begin{aligned} \|f(0) - f(1)\| = \Bigg\| \int_0^1 \mathrm{d}s \sum_{k=1}^{m} \left(\prod_{l=1}^{k-1} \mathrm{e}^{-\mathrm{i}\,(\mathrm{D}+s\tilde{\mathrm{D}}_l)\tau/m} \right) \\ \times \left(\int_0^1 \mathrm{d}\hat{s} \mathrm{e}^{-\mathrm{i}\,(\mathrm{D}+s\tilde{\mathrm{D}}_k)(\tau/m)\hat{s}} \left(-\mathrm{i}\,\tilde{\mathrm{D}}_k \frac{\tau}{m} \right) \mathrm{e}^{-\mathrm{i}\,(\mathrm{D}+s\tilde{\mathrm{D}}_k)(\tau/m)(1-\hat{s})} \right) \\ \times \left(\prod_{l=k+1}^{m} \mathrm{e}^{-\mathrm{i}\,(\mathrm{D}+s\tilde{\mathrm{D}}_l)\tau/m} \right) \Bigg\| \\ \leqslant \int_0^1 \mathrm{d}s \int_0^1 \mathrm{d}\hat{s} \sum_{k=1}^{m} \Bigg\| \left(\prod_{l=1}^{k-1} \mathrm{e}^{-\mathrm{i}\,(\mathrm{D}+s\tilde{\mathrm{D}}_l)\tau/m} \right) \\ \times \left(\mathrm{e}^{-\mathrm{i}\,(\mathrm{D}+s\tilde{\mathrm{D}}_k)(\tau/m)\hat{s}} \left(-\mathrm{i}\,\tilde{\mathrm{D}}_k \frac{\tau}{m} \right) \mathrm{e}^{-\mathrm{i}\,(\mathrm{D}+s\tilde{\mathrm{D}}_k)(\tau/m)(1-\hat{s})} \right) \\ \times \left(\prod_{l=k+1}^{m} \mathrm{e}^{-\mathrm{i}\,(\mathrm{D}+s\tilde{\mathrm{D}}_l)\tau/m} \right) \Bigg\| \\ \leqslant \int_0^1 \mathrm{d}s \int_0^1 \mathrm{d}\hat{s} \sum_{k=1}^{m} \|\tilde{\mathrm{D}}_k \tau/m\| = \sum_{k=1}^{m} \|\tilde{\mathrm{D}}_k\| \, \frac{\tau}{m}. \end{aligned} \tag{A.5}$$

Note that if $\|\tilde{\mathrm{D}}_l\| \leqslant c$, with c a constant then

$$\left\| \mathrm{e}^{-\mathrm{i}\,\mathrm{D}\tau} - \prod_{l=1}^{m} \mathrm{e}^{-\mathrm{i}\,(\mathrm{D}+\tilde{\mathrm{D}}_l)\tau/m} \right\| \leqslant c\tau, \tag{A.6}$$

independent of m.

Appendix B. Postselection error

We discuss the approximation in equation (11). By construction we have

$$\left\| \sum_j \beta_j \tilde{\lambda}_j(\mathbf{D})|u_j(\mathbf{D})\rangle - \mathbf{D}|\mathbf{x}^{(t)}\rangle \right\| \leqslant \epsilon_D. \tag{B.1}$$

We also have the upper bound for the difference in normalizations

$$|\tilde{C}_D^{(t+1)} - C_D^{(t+1)}| = \mathcal{O}(\eta\epsilon_D). \tag{B.2}$$

Similar to equation (14) we find the lower bound

$$\tilde{C}_D^{(t+1)} = \Omega\left(\frac{1}{2} - \eta\epsilon_D\right). \tag{B.3}$$

Thus we have (similar to [42], lemma 7)

$$\begin{aligned} &\left\| \frac{1}{\tilde{C}_D^{(t+1)}}\left(|\mathbf{x}^{(t)}\rangle - \eta\sum_j \beta_j \tilde{\lambda}_j(\mathbf{D})|u_j(\mathbf{D})\rangle\right) - \frac{1}{C_D^{(t+1)}}(|\mathbf{x}^{(t)}\rangle - \eta\mathbf{D}|\mathbf{x}^{(t)}\rangle) \right\| \\ &= \mathcal{O}\left(\frac{\eta\epsilon_D}{\tilde{C}_D^{(t+1)}}\right) = \mathcal{O}\left(\frac{\eta\epsilon_D}{1 - 2\eta\epsilon_D}\right) = \mathcal{O}(\eta\epsilon_D). \end{aligned} \tag{B.4}$$

The last step holds if $\eta\epsilon_D < 1/2$. From result 1, this condition is easily satisfied from $\epsilon_D < 1$, $\eta < 1/(2p\Lambda_A)$, and p, $\Lambda_A \geqslant 1$.

ORCID iDs

Maria Schuld https://orcid.org/0000-0001-8626-168X

References

[1] Sra S, Nowozin S and Wright S J 2012 *Optimization for Machine Learning* (Cambridge, MA: MIT Press)
[2] Martens J 2010 Deep learning via hessian-free optimization *Proc. 27th Int. Conf. on Machine Learning (ICML-10)* pp 735–42
[3] Boyd S and Vandenberghe L 2004 *Convex Optimization* (Cambridge: Cambridge University Press)
[4] Nocedal J and Wright S 2006 *Numerical Optimization* (Berlin: Springer)
[5] Dürr C and Hoyer P 1996 A quantum algorithm for finding the minimum arXiv:quant-ph/9607014
[6] Farhi E and Harrow A W 2016 Quantum supremacy through the quantum approximate optimization algorithm arXiv:1602.07674
[7] Farhi E, Goldstone J, Gutmann S and Sipser M 2000 Quantum computation by adiabatic evolution arXiv:quant-ph/0001106 MIT-CTP-2936
[8] Denchev V, Ding N, Neven H and Vishwanathan S 2012 Robust classification with adiabatic quantum optimization *Proc. 29th Int. Conf. on Machine Learning (ICML-12)* (Edinburgh: Omnipress) pp 1003–10
[9] Neven H, Denchev V S, Rose G and Macready W G 2009 Training a large scale classifier with the quantum adiabatic algorithm arXiv:0912.0779
[10] Benedetti M, Realpe-Gómez J, Biswas R and Perdomo-Ortiz A 2016 *Phys. Rev.* A **94** 022308
[11] Wiebe N, Braun D and Lloyd S 2012 Quantum algorithm for data fitting *Phys. Rev. Lett.* **109** 050505
[12] Schuld M, Sinayskiy I and Petruccione F 2016 Prediction by linear regression on a quantum computer *Phys. Rev.* A **94** 022342
[13] Rebentrost P, Mohseni M and Lloyd S 2014 Quantum support vector machine for big data classification *Phys. Rev. Lett.* **113** 130503
[14] Harrow A W, Hassidim A and Lloyd S 2009 Quantum algorithm for linear systems of equations *Phys. Rev. Lett.* **103** 150502
[15] Peruzzo A, McClean J, Shadbolt P, Yung M-H, Zhou X-Q, Love P J, Aspuru-Guzik A and O'brien J L 2014 A variational eigenvalue solver on a photonic quantum processor *Nat. Commun.* **5** 4213
[16] Farhi E, Goldstone J and Gutmann S 2014 A quantum approximate optimization algorithm arXiv:1411.4028 [quant-ph]
[17] Lloyd S, Mohseni M and Rebentrost P 2014 Quantum principal component analysis *Nat. Phys.* **10** 631–3
[18] Berry D W and Childs A M 2012 Black-box Hamiltonian simulation and unitary implementation *Quantum Info. Comput.* **12** 29–62
[19] Jordan S P 2005 Fast quantum algorithm for numerical gradient estimation *Phys. Rev. Lett.* **95** 050501
[20] Kerenidis I and Prakash A 2017 Quantum gradient descent for linear systems and least squares arXiv:1704.04992
[21] Gilyén A, Arunachalam S and Wiebe N 2017 Optimizing quantum optimization algorithms via faster quantum gradient computation *Proc. 30th ACM-SIAM Symp. on Discrete Algorithms (SODA 2019)* pp 1425–44
[22] He S, Li Z and Zhang S 2010 Approximation algorithms for homogeneous polynomial optimization with quadratic constraints *Math. Program.* **125** 353–83
[23] Goldstein A A 1964 Convex programming in hilbert space *Bull. Am. Math. Soc.* **70** 709–10
[24] Levitin E S and Polyak B T 1966 Constrained minimization methods *USSR Comput. Math. Math. Phys.* **6** 1–50
[25] Giovannetti V, Lloyd S and Maccone L 2008 Quantum random access memory *Phys. Rev. Lett.* **100** 160501
[26] Giovannetti V, Lloyd S and Maccone L 2008 Architectures for a quantum random access memory *Phys. Rev.* A **78** 052310
[27] De Martini F, Giovannetti V, Lloyd S, Maccone L, Nagali E, Sansoni L and Sciarrino F 2009 Experimental quantum private queries with linear optics *Phys. Rev.* A **80** 010302
[28] Grover L and Rudolph T 2002 Creating superpositions that correspond to efficiently integrable probability distributions arXiv:quant-ph/0208112

[29] Soklakov A N and Schack R 2006 Efficient state preparation for a register of quantum bits *Phys. Rev.* A **73** 012307
[30] Low G H and Chuang I L 2017 Optimal Hamiltonian simulation by quantum signal processing *Phys. Rev. Lett.* **118** 010501
[31] Childs A M, Maslov D, Nam Y, Ross N J and Su Y 2018 Toward the first quantum simulation with quantum speedup *Proc. National Academy of Sciences* vol 115, pp 9456–61
[32] Hinton G E, Osindero S and Teh Y-W 2006 A fast learning algorithm for deep belief nets *Neural Comput.* **18** 1527–54
[33] Bengio Y 2009 Learning deep architectures for ai *Found. Trends® Mach. Learn.* **2** 1–127
[34] Kimmel S, Lin C Y-Y, Low G H, Ozols M and Yoder T J 2017 Hamiltonian simulation with optimal sample complexity *NPJ Quantum Inf.* **3** 13
[35] Bertsekas D P 1982 Projected newton methods for optimization problems with simple constraints *SIAM J. Control Optim.* **20** 221
[36] Lee J D, Sun Y and Saunders M A 2012 Proximal newton-type methods for convex optimization *Proc. 25th Int. Conf. on Neural Information Processing Systems—Vol 1, NIPS'12 (Curran Associates Inc., USA)* pp 827–35
[37] Childs A M, Kothari R and Somma R D 2017 Quantum linear systems algorithm with exponentially improved dependence on precision *SIAM J. Comput.* **46** 1920–50
[38] Brassard G, Hoyer P, Mosca M and Tapp A 2002 Quantum amplitude amplification and estimation *Contemp. Math.* **305** 53–74
[39] Ambainis A 2012 Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations *29th Int. Symp. Theoretical Aspects of Computer Science (STACS 2012)* vol 14 (Dagstuhl: Schloss) pp 636–47
[40] Dauphin Y N, Pascanu R, Gulcehre C, Cho K, Ganguli S and Bengio Y 2014 Identifying and attacking the saddle point problem in high-dimensional non-convex optimization *Adv. Neural Inf. Process. Syst.* **27** 2933–41
[41] Haber H E 2018 Notes on the matrix exponential and logarithm
[42] van Apeldoorn J, Gilyén A, Gribling S and de Wolf R 2017 Quantum sdp-solvers: Better upper and lower bounds *58th IEEE Symposium on Foundations of Computer Science (FOCS 2017)* pp 403–14