Fast Quantum Algorithm for Numerical Gradient Estimation

Stephen P. Jordan*

Physics Department, MIT, 77 Massachusetts Avenue, Cambridge, Massachusetts 02139, USA

(Received 25 May 2004; revised manuscript received 3 January 2005; published 28 July 2005)

Given a black box for f, a smooth real scalar function of d real variables, one wants to estimate ∇f at a given point with n bits of precision. On a classical computer this requires a minimum of $d + 1$ black box queries, whereas on a quantum computer it requires only one query regardless of d. The number of bits of precision to which f must be evaluated matches the classical requirement in the limit of large n.

DOI: 10.1103/PhysRevLett.95.050501 PACS numbers: 03.67.Lx

We investigate the query complexity of numerically estimating the gradient of a black box function f: $R^d \rightarrow R$ at a given point. We find that gradients can be estimated on a quantum computer using a single black box query whereas we require at least $d + 1$ queries classically. The algorithm which achieves this can be viewed as a generalization of the Bernstein-Vazirani [1] algorithm, which has been described in other contexts [2–5]. The black box in this algorithm was always previously described as evaluating a function over the integers rather than approximating a continuous function with finite precision. In [2], the question as to whether the algorithm could be adapted for any task of practical interest was presented as an open problem, which this paper resolves by showing a speedup for gradient estimation, which is a fundamental operation in many numerical calculations.

For many numerical calculations, black box query complexity is a natural measure of algorithmic efficiency. For example, function evaluations are frequently the most time consuming part of solving numerical optimization problems. An efficient optimization algorithm is therefore one which uses as few function evaluations as possible [6].

The black box that we consider evaluates some smooth function f: $R^d \rightarrow R$. It does so with finite precision determined by the number of bits used to represent $\mathbf{x}$ and $f(\mathbf{x})$. For simplicity we will discuss gradient estimation at the origin, since the gradient at other points can be obtained by trivially redefining f. To estimate ∇f, in either the classical or quantum case, one samples f over a region sufficiently small that expanding f to first order is a good approximation: $f(\mathbf{x}) \simeq f(\mathbf{0}) + \mathbf{x} \cdot \nabla f$. In the quantum case, the evaluations of f at the different sample points can be done in superposition, necessitating only a single evaluation of f.

Classically, to estimate ∇f in d dimensions we need to evaluate f at least $d + 1$ times. Suppose the quadratic and higher terms in f can be neglected, then $f(\mathbf{x}) = f(\mathbf{0}) + \mathbf{x} \cdot \nabla f$ for small $\mathbf{x}$. Each evaluation of f at a given point $\mathbf{x}_i$ gives us a linear equation of the form $f(\mathbf{0}) + \mathbf{x}_i \cdot \nabla f = f_i$. We have $d + 1$ unknowns [the d components of ∇f plus $f(\mathbf{0})$], thus evaluating f at fewer than $d + 1$ distinct points would leave the system of linear equations underdetermined. A natural choice of the $d + 1$ points which allows us to solve this system is $\mathbf{0}$ and the d points displaced from $\mathbf{0}$ by a small amount l along each axis, as shown in Fig. 1. The quadratic and higher terms in f contribute an error to the gradient estimate which shrinks linearly with l.

In practice, it may be desirable in the classical gradient estimation algorithm to perform the function evaluations displaced by $\pm l/2$ from the origin along each dimension. In this case $2d$ function evaluations are required instead of $d + 1$. $\partial f/\partial x_1$ will then be given by $[f(+l/2, 0\ldots) - f(-l/2, 0\ldots)]/l$, and similarly for the other partial derivatives. Inserting the Taylor expansion for f into this expression shows that the terms quadratic in $\mathbf{x}$ cancel, leaving an error of order l^2 and higher, which is the advantage of this approach.

Now we consider the quantum case. The black box takes as its input d binary strings, each of length n, along with n_o ancilla qubits all initialized to zero. The black box writes its output into the ancilla qubits using addition modulo $N_o \equiv 2^{n_o}$ and preserves the input qubits. This is a standard technique for making any classical function reversible, which it must be for a quantum computer to implement it.

The qubit strings which form the input to the black box represent the components of $\mathbf{x}$ in fixed-point notation. We will choose the components of $\mathbf{x}$ to be between $-l/2$ and $l/2$ where l is sufficiently small so that f is approximately linear throughout this domain. Just as in the classical case we need some *a priori* knowledge of how small to make l. To convert between the values of $\mathbf{x} \in R^d$ and the positive integers $\boldsymbol{\delta}$ represented by the qubit strings which are input to the black box we use $\mathbf{x} = \frac{l}{N}(\boldsymbol{\delta} - \frac{\mathbf{N}}{2})$ where the components of $\boldsymbol{\delta}$ are n-bit integers (0 to $N \equiv 2^n$) and $\mathbf{N}$ is the

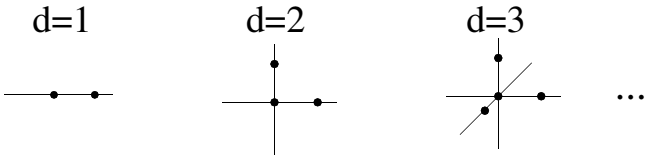

FIG. 1. Classically, we could estimate ∇f using $d + 1$ function evaluations according to $\frac{\partial f}{\partial x_i} \simeq [f(\mathbf{x} + l\hat{e}_i) - f(\mathbf{x})]/l$ where $\hat{e}_i$ is the ith normalized basis vector.

d-dimensional vector $(N, N, N, \ldots)$ which centers the region being sampled on the origin.

Similarly, the output of the black box represents the value of f in fixed-point notation. For maximal precision while using minimal qubits, one must have an order of magnitude estimate of the range of f in the domain of interest. In our case the range of f is roughly from $f(\mathbf{0}) - Ml/2$ to $f(\mathbf{0}) + Ml/2$ where M is the largest magnitude of any first partial derivative of f. The integer which gets added modulo N_o to the output register will be related to the real value of f by

$$\left\lceil \frac{NN_o}{ml} f(\mathbf{x}) \right\rfloor \tag{1}$$

where m is our estimate of M and $\lceil\rfloor$ denotes rounding to the nearest integer.

The first step in the quantum gradient estimation algorithm is to create a uniform superposition over inputs by performing the Hadamard transform on the input registers. Next, create a ''plane wave'' state in the output register by writing 1 into the output register and then performing the inverse Fourier transform on it. This yields

$$\frac{1}{\sqrt{N^d N_o}} \sum_{\delta_1=0}^{N-1} \sum_{\delta_2=0}^{N-1} \cdots \sum_{\delta_d=0}^{N-1} |\delta_1\rangle \ldots |\delta_d\rangle \sum_{a=0}^{N_o-1} e^{i2\pi a/N_o} |a\rangle,$$

or in vector notation

$$= \frac{1}{\sqrt{N^d N_o}} \sum_{\boldsymbol{\delta}} |\boldsymbol{\delta}\rangle \sum_a e^{i2\pi a/N_o} |a\rangle.$$

Next, use the black box to compute f and add it modulo N_0 into the output register. The output register is in an eigenstate of addition modulo N_0. The eigenvalue corresponding to addition of x is $e^{i2\pi x/N_o}$. Thus by writing into the output register via modular addition, we obtain a phase proportional to f. This technique is sometimes called phase kickback. The resulting state is [7]

$$\frac{1}{\sqrt{N^d N_o}} \sum_{\boldsymbol{\delta}} e^{i2\pi(N/ml) f[(l/n)(\boldsymbol{\delta} - \mathbf{N}/2)]} |\boldsymbol{\delta}\rangle \sum_a e^{i2\pi a/N_o} |a\rangle.$$

For sufficiently small l,

$$\approx \frac{1}{\sqrt{N^d N_o}} \sum_{\boldsymbol{\delta}} e^{(i2\pi(N/ml)\{f(\mathbf{0}) + (l/N)[\boldsymbol{\delta} - (\mathbf{N}/2)] \cdot \nabla f\})} |\boldsymbol{\delta}\rangle \times \sum_a e^{i2\pi a/N_o} |a\rangle.$$

Writing out the vector components, and ignoring global phase, the input registers are now approximately in the state

$$= \frac{1}{\sqrt{N^d}} \sum_{\delta_1 \ldots \delta_d} e^{i(2\pi/m)[\delta_1(\partial f/\partial x_1) + \delta_2(\partial f/\partial x_2) + \cdots + \delta_d(\partial f/\partial x_d)]} \times |\delta_1\rangle |\delta_2\rangle \ldots |\delta_d\rangle.$$

This is a product state:

$$= \frac{1}{\sqrt{N^d}} \times \left(\sum_{\delta_1} e^{i(2\pi/m)\delta_1(\partial f/\partial x_1)} |\delta_1\rangle \right) \ldots \left(\sum_{\delta_d} e^{i(2\pi/m)\delta_d(\partial f/\partial x_d)} |\delta_d\rangle \right).$$

Fourier transform each of the registers, obtaining

$$\left| \frac{N}{m} \frac{\partial f}{\partial x_1} \right\rangle \left| \frac{N}{m} \frac{\partial f}{\partial x_2} \right\rangle \ldots \left| \frac{N}{m} \frac{\partial f}{\partial x_d} \right\rangle.$$

Then simply measure in the computational basis to obtain the components of ∇f with n bits of precision. Because f will, in general, not be perfectly linear, even over a small region, there also will be nonzero amplitude to measure other values close to the exact gradient, as will be discussed later.

Normally, the quantum Fourier transform is thought of as mapping the discrete plane wave states to the computational basis states:

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N} e^{2\pi i jk/N} |j\rangle \rightarrow |k\rangle$$

where $0 < k < N$. However, negative k is also easily dealt with, since

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N} e^{-2\pi i j|k|/N} \rightarrow |N - |k|\rangle.$$

Thus negative components of ∇f pose no difficulties for the quantum gradient estimation algorithm provided that bounds for the values of the components are known, which is a requirement for any algorithm using fixed-point arithmetic.

In general the number of bits of precision necessary to represent a set of values is equal to $\log_2(r/\delta)$, where r is the range of values, and δ is the smallest difference in values one wishes to distinguish. Thus for classical gradient estimation with n bits of precision, one needs to evaluate f to

$$\log_2 \left\lceil \frac{\max f - \min f}{ml/2^n} \right\rceil \tag{2}$$

bits of precision.

An important property of the quantum Fourier transform is that it can correctly distinguish between exponentially many discrete plane wave states with high probability without requiring the phases to be exponentially precise [8]. It is not hard to show that if each phase is accurate to within θ then the inner product between the ideal state and the actual state is at least $\cos\theta$, and therefore the algorithm will still succeed with probability at least $\cos^2\theta$.

As shown earlier, the phase acquired by ''kickback'' is equal to $\frac{2\pi N}{ml} f$, and therefore, for the phase to be accurate to within $\pm\theta$, f must be evaluated to within $\pm \frac{ml}{2\pi N} \theta$. Thus,

recalling that $N = 2^n$,

$$n_o = \log_2\left[\frac{\max f - \min f}{(ml/2^n)(\theta/2\pi)}\right]. \quad (3)$$

As an example, if $\theta = \pi/8$, then the algorithm will behave exactly as in the idealized case with at least 85% probability, and n_o will exceed the classically required precision by 4 bits, for a given value of l. l also differs between the quantum and classical cases, as will be discussed later. Thus n_o differs from the classically required precision only by an additive constant which depends on θ and l. Because the classical and quantum precision requirements grow linearly with n, this difference becomes negligible in the limit of large n.

The only approximation made in the description of the quantum gradient estimation algorithm was expanding f to first order. Therefore the lowest order error term will be due to the quadratic part of f. The behavior of the algorithm in the presence of such a quadratic term provides an idea of its robustness. Furthermore, in order to minimize the number of bits of precision to which f must be evaluated, l should be chosen as large as possible subject to the constraint that f be locally linear. The analysis of the quadratic term provides a more precise description of this constraint.

The series of quantum Fourier transforms on different registers can be thought of as a single d-dimensional quantum Fourier transform. Including the quadratic term, the state which this Fourier transform is acting on has amplitudes

$$a(\boldsymbol{\delta}) = \frac{1}{N^{d/2}} \exp\left[i2\pi\left(\frac{1}{m}\boldsymbol{\delta}\cdot\nabla f + \frac{l}{2mN}\boldsymbol{\delta}^T H\boldsymbol{\delta}\right)\right],$$

where H is the Hessian matrix of f. After the Fourier transform, the amplitudes should peak around the correct value of ∇f. Here we are interested in the width of the peak, which should not be affected by ∇f, so for simplicity ∇f will be set to $\mathbf{0}$. The Fourier transform will yield amplitudes of [9]

$$\tilde{a}(\mathbf{k}) = \frac{1}{N^d}\sum_{\boldsymbol{\delta}} \exp\left[i2\pi\left(\frac{l}{2mN}\boldsymbol{\delta}^T H\boldsymbol{\delta} - \frac{1}{N}\mathbf{k}\cdot\boldsymbol{\delta}\right)\right].$$

Ignoring global phase and doing a change of variables ($\mathbf{u} = \boldsymbol{\delta}/N$),

$$\approx \int_{-1/2}^{1/2}\cdots\int_{-1/2}^{1/2} \exp\left[i2\pi\left(\frac{Nl}{2m}\mathbf{u}^T H\mathbf{u} - \mathbf{k}\cdot\mathbf{u}\right)\right]\mathrm{d}^d\mathbf{u}.$$

This integral can be approximated using the method of stationary phase. The gradient of the phase of the integrand is $\nabla\phi = \frac{Nl}{2m}(H^T + H)\mathbf{u} - \mathbf{k}$ but Hessians are symmetric, so $\nabla\phi = \frac{Nl}{m}H\mathbf{u} - \mathbf{k}$. Thus (again ignoring global phase),

$$\tilde{a}(\mathbf{k}) \approx \begin{cases} \sqrt{\frac{1}{\det(\frac{Nl}{m}H)}} & \text{if } \exists\mathbf{u}\in C \text{s.t.} \frac{Nl}{m}H\mathbf{u} - \mathbf{k} = \mathbf{0} \\ 0 & \text{otherwise} \end{cases}$$

where C is the region $-1/2 < u_i < 1/2 \quad \forall\ i$. In this approximation, the peak is simply a region of uniform amplitude, with zero amplitude elsewhere. Geometrically, the linear transformation $\frac{Nl}{m}H$ applied to the d-dimensional unit hypercube yields this region.

Since we have set $\nabla f = \mathbf{0}$, the variance of $\frac{N}{m}\frac{\partial f}{\partial x_i}$ will be

$$\sigma_i^2 = \frac{1}{\det A}\int_D k_i^2 d^d\mathbf{k}$$
$$\text{where } A = \frac{Nl}{m}H$$

and D is the region of nonzero amplitude. Doing a change of variables with A as the Jacobian,

$$\sigma_i^2 = \frac{1}{\det A}\int_C (A\mathbf{k}')_i^2 \det A d^d\mathbf{k}'$$

where C is again the unit hypercube centered at the origin. In components,

$$\sigma_i^2 = \int_C\left(\sum_j A_{ij}k_j'\right)^2 d^d\mathbf{k}'.$$

The expectation values on a hypercube of uniform probability are $\langle k_i k_j\rangle = \frac{1}{12}\delta_{ij}$, thus

$$\sigma_i^2 = \frac{1}{12}\sum_j A_{ij}^2 = \frac{N^2l^2}{12m^2}\sum_j\left(\frac{\partial^2 f}{\partial x_i\partial x_j}\right)^2. \quad (4)$$

This quadratic dependence on N is just as expected since, at the end of the computation, the register that we are measuring is intended to contain $\frac{N}{m}\frac{\partial f}{\partial x_i}$. Therefore the uncertainty in $\partial f/\partial x_i$ is approximately

$$\frac{l}{2\sqrt{3}}\sqrt{\sum_j\left(\frac{\partial^2 f}{\partial x_i\partial x_j}\right)^2}$$

independent of N. In the classical algorithm which uses $2d$ function evaluations, the cubic term introduces an error of $\sigma \sim \frac{l^2}{24}D_3$ where D_3 is the typical [10] magnitude of third partial derivatives of f. If the 2nd partial derivatives of f have a magnitude of approximately D_2 then the typical uncertainty in the quantum case will be $\sigma \sim \frac{lD_2\sqrt{d}}{2\sqrt{3}}$. To

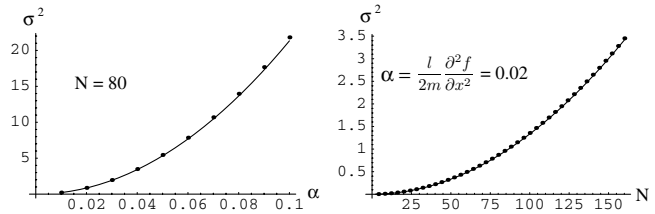

FIG. 2. Comparison between error estimates obtained in the stationary phase approximation (solid line) and numerical results (points) for the one dimensional case. On the right the 2nd derivative remains constant ($\alpha = 0.02$) and the number of bits (N) varies, and vice versa on the left.

obtain a given uncertainty σ, l must be chosen according to

$$l \sim \begin{cases} 2\sqrt{\frac{6\sigma}{D_3}} & \text{classical} \\ \frac{2\sqrt{3}\sigma}{D_2\sqrt{d}} & \text{quantum} \end{cases}.$$

Recalling Eqs. (2) and (3), the number of bits of precision to which f must be evaluated depends logarithmically on l. However, in the limit of large n, the number of bits will match the classical requirement.

The level of accuracy of the stationary phase approximation can be assessed by comparison to numerical solutions of example cases. In one dimension, Eq. (4) reduces to $\sigma^2 = \frac{\alpha^2 N^2}{3}$ where $\alpha = \frac{l}{2m}\frac{\partial^2 f}{\partial x^2}$. Figure 2 displays the close agreement between numerical results and the analytical solution obtained using stationary phase.

A two dimensional example provides a nontrivial test of the stationary phase method's prediction of the peak shape. If the Hessian is such that

$$\frac{N}{m}H = 0.1\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

then, according to the stationary phase approximation, the peak should be a square of side length $\frac{\sqrt{2}}{10}l$ with a 45° rotation. This is in reasonable agreement with the numerical result, as shown in Fig. 3.

Because this algorithm requires only one black box query, one might expect that it could be run recursively to efficiently obtain higher derivatives. Another instance of the same algorithm would serve as the black box. However, the algorithm differs from the black box in that the black box has scalar output which it adds modulo N_o into the output register, and it does not incur any input-dependent global phase. An additive scalar output can be obtained by minor modification to this algorithm, but the most straightforward techniques for eliminating the global phase require an additional black box query, thus necessitating 2^n queries for the evaluation of an nth partial derivative, just as in the classical case.

The problem of global phase when recursing quantum algorithms as well as the difficulties inherent in recursing approximate or probabilistic algorithms are not specific to gradient finding but are instead fairly general.

Efficient gradient estimation may be useful, for example, in some optimization and rootfinding algorithms. Furthermore, upon discretization, the problem of minimizing a functional is converted into the problem of minimizing a function of many variables, which might benefit from gradient descent techniques. A speedup in the minimization of functionals may in turn enable more efficient solution of partial differential equations via the Euler-Lagrange equation. The analysis of the advantage which this technique can provide in quantum numerical algorithms remains open for further research.

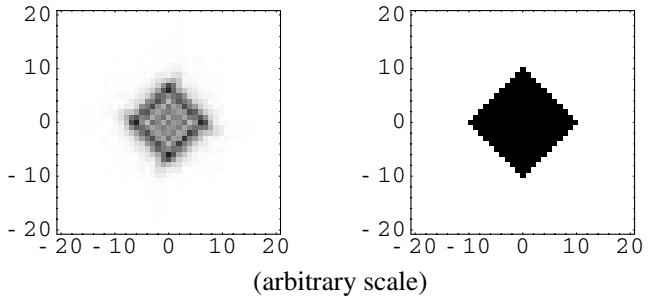

FIG. 3. On the left, the probability, as numerically calculated, is shown. The areas of highest probability appear darkest. On the right, the region of nonzero probability, as calculated in the stationary phase approximation, is shaded in black.

The author thanks P. Shor, E. Farhi, L. Grover, J. Traub, and M. Rudner for useful discussions and MIT and ARO's QuaCGR program for partial support.

*Email address: sjordan@mit.edu

[1] E. Bernstein and U. Vazirani, *Proceedings of the 25th ACM Symposium on the Theory of Computing* (ACM Press, New York, 1993), pp. 11–20.
[2] M. Mosca, Ph.D. thesis, University of Oxford, 1999.
[3] P. Høyer, Phys. Rev. A **59**, 3280 (1999).
[4] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, Proc. R. Soc. A **454**, 339 (1998).
[5] C. H. Bennett, G. Brassard, P. Høyer, and U. Vazirani, SIAM J. Comput. **26**, 1510 (1997).
[6] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C* (Cambridge University Press, Cambridge, England, 1992), 2nd ed..
[7] Equation (1) was chosen such that our output value routinely exceeds N_o, but here we see that the information thrown away when we take mod N_o only corresponds an irrelevant multiple of 2π added to the phase.
[8] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, England, 2000), (See exercise 5.6).
[9] $\boldsymbol{\delta}$ here really represents $\boldsymbol{\delta} - \mathbf{N}/2$.
[10] Alternatively, we can define D_3 and D_2 as the largest 2nd and 3rd partial derivatives of f to obtain a worst case requirement on l.