Quantum Newton’s method for solving system of nonlinear algebraic equations

Cheng Xue,1 Yu-Chun Wu,1,2,* and Guo-Ping Guo1,2,3

1CAS Key Laboratory of Quantum Information, University of Science and Technology of China, Hefei, Anhui 230026, P. R. China
2Institute of Artificial Intelligence, Hefei Comprehensive National Science Center Hefei, Anhui 230026, P. R. China
3Origin Quantum Computing, Hefei, Anhui 230026, P. R. China

While quantum computing provides an exponential advantage in solving system of linear equations, there is little work to solve system of nonlinear equations with quantum computing. We propose quantum Newton’s method (QNM) for solving N-dimensional system of nonlinear equations based on Newton’s method. In QNM, we solve the system of linear equations in each iteration of Newton’s method with quantum linear system solver. We use a specific quantum data structure and l∞ tomography with sample error εs to implement the classical-quantum data conversion process between the two iterations of QNM, thereby constructing the whole process of QNM. The complexity of QNM in each iteration is \(O(\log^4 N/\epsilon_s^2)\). Through numerical simulation, we find that when \(\epsilon_s >> 1/\sqrt{N}\), QNM is still effective, so the complexity of QNM is sublinear with \(N\), which provides quantum advantage compared with the optimal classical algorithm.

I. INTRODUCTION

The system of nonlinear equations appears in many fields, such as nonlinear finite element analysis[1], nonlinear dynamics[2], nonlinear programming[3], etc. In general, the analytical solutions of system of nonlinear equations cannot be obtained effectively. Numerical methods are often used to solve system of nonlinear equations. Newton’s method is a basic method for solving system of nonlinear equations[4, 5]. However, the complexity of solving a system of nonlinear equations with Newton’s method grows polynomially with the dimension of the equations[5], using Newton’s method to solve high-dimensional nonlinear equations is intractable. It is important to develop more efficient algorithms for solving system of nonlinear equations.

Quantum computing provides a promising way to speed up the solution of system of nonlinear equations. In recent years there have been many quantum algorithms developed to solve various equations, such as system of linear equations[6–8], system of linear differential equations[9–21] and nonlinear differential equations[22–25].

However, there is little work for the solution of system of nonlinear equations. Qian et al.[26] proposed a quantum algorithm for solving system of nonlinear equations, their work is based on Grover algorithm and only has square acceleration. Rebentrost et al.[27] proposed a quantum Newton’s method for constrained polynomial optimization, the complexity of their method grows exponentially as iteration times increases.

In our work, we propose a quantum Newton’s method (QNM) for the solution of system of nonlinear equations. QNM is an iteration method, in each iteration, we solve a system of linear equations with quantum linear system solver (QLSS). Between two iterations, there are some classical-quantum data conversions. We realize the conversions by using a quantum data structure and \(l_\infty\) tomography[28] with sample error \(\epsilon_s\).

The complexity of QNM in each iteration is \(O(\log^4 N/\epsilon_s^2)\). Through numerical simulation, we found that when \(\epsilon_s >> 1/\sqrt{N}\), QNM is still effective, so the complexity of QNM is sublinear with \(N\). Compared with the best classical algorithm, QNM has a significant acceleration. The complexity of QNM grows linearly with iteration times, which is better than the work proposed in [27].

This paper is organized as follows. We introduce Newton’s method in Sec. II. Then the details of QNM are discussed in Sec. III and we analyze the complexity of QNM in Sec. IV. Sec. V gives some applications of QNM and some numerical simulation to verify the effectiveness of QNM. Finally, we summarize our work and propose some future research directions in Sec. VI.

II. NEWTON’S METHOD

Newton’s method is an iterative method for solving system of nonlinear equations. It obtains solution of the system of nonlinear equations by iteratively solving system of linear equations.

The system of nonlinear equations is defined as:

\[ F(x) = 0, \tag{1} \]

where

\[ F(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_N(x) \end{bmatrix}, x = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}, 0 = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{2} \]

each \(f_i(x)\) is a real-valued nonlinear function that maps \(x \in \mathbb{R}^N\) to \(\mathbb{R}\). We use \(x^*\) to represent the solution of Eq.(1). The Jacobi matrix of \(F(x)\) is defined as:

\[ F'(x) = (\partial_j f_i(x))_{N \times N} = (a_{ij})_{N \times N}, \tag{3} \]
The process of Newton’s method is shown in Algorithm 1.

**Algorithm 1** Newton’s method

**Input:** $F(x)$, initial vector $x^0$ and accuracy error $\epsilon$.

**Output:** Solution $x^*$.

1: Assuming $k$ iterations have been performed, $x^k$ and $F(x^k)$ have been found. Compute Jacobi matrix $F'(x^k) = A_k$, and define $b_k = -F(x^k)$;
2: Solve the system linear of equations:
   \[ A_k \Delta x^k = b^k. \]  
(4)
3: Compute $x^{k+1} = x^k + \Delta x^k$ and $F(x^{k+1})$.
4: If $||F(x^{k+1})|| \leq \epsilon$, we have $x^* = x^{k+1}$ and go to step 5. Otherwise, $k + 1 \rightarrow k$, $x^{k+1} \rightarrow x^k$, $F(x^{k+1}) \rightarrow F(x^k)$, go to step 1.
5: Return $x^*$.
6: End.

### III. QNM

In our work, we only consider sparse system of non-linear equations, the Jacobi matrix $F'(x)$ is a $d$-sparse matrix, which means $F'(x)$ has at most $d$ non-zero elements in any row or column. Throughout this paper, $d$ represents the sparsity of $F'(x)$. In QNM, we need some oracles of Eq.(1). Suppose we are given oracles $O_{f1}$ and $O_{f2}$ that provide the information of Eq.(1), $O_{f1}$ and $O_{f2}$ are defined as:

\[ O_{f1}|i|j) = |i, f(i, j)), i = 1, 2, ..., N, j = 1, 2, ..., d \]  
(5)

\[ O_{f2}|i)(x|0) = |i)|x(j)|f_i(x(i)), i = 1, 2, ..., N. \]  
(6)

where $f(i, j)$ is the subscript of the $j$-th variable in $f_j(x)$, and $x(i)$ represents the related $x_j$ in $f_j(x)$. The construction of $O_{f1}$ and $O_{f2}$ are related to specific problems. In many specific problems, both $O_{f1}$ and $O_{f2}$ can be implemented with quantum arithmetic. In Sec.V A, we give some specific examples.

QNM is a quantum version of Newton’s method. In each iteration of Newton’s method, we solve a system of linear equations represented by Eq.(4). In QNM, we implement the process of solving the system of linear equations with QLSS, and give the conversion process of classical-quantum data during each iteration. The overall block diagram of QNM is shown in Fig.1. Each iteration of QNM is divided into three steps: preprocessing, processing and postprocessing. The details of these three steps are introduced in Sec.III B, Sec.III C and Sec.III D. During the execution of these 3 steps, we use a quantum data structure $M_F$. Let us introduce $M_F$ first.

**A. Quantum data structure**

In QNM, we use $M_F$ to implement some sub-processes. The structure of $M_F$ is inspired by previous work[29–31]. $M_F$ is a binary tree structure. In each leaf node, we store $x_i$ and $f_i(x)$. In other nodes, we store $d_{ijx}$ and $d_{ijy}$, which are defined as:

\[ d_{ijx} = \frac{1}{(j+1) \times 2^{n-i} - 1} \sum_{k=j \times 2^{n-i}}^{(j+1) \times 2^{n-i} - 1} x_k^2 \]  
\[ d_{ijy} = \frac{1}{(j+1) \times 2^{n-i} - 1} \sum_{k=j \times 2^{n-i}}^{(j+1) \times 2^{n-i} - 1} f_k^2(x) \]  
(7)

where $i, j$ represent the $j$-th node of the $i$-th layer, $n = \lceil \log N \rceil$. In the root node, we store $d_{00x} = ||x||$ and $d_{00y} = ||F(x)||$. Fig. 2 shows the structure of $M_F$ with
dimension \(N = 4\). Given an \(N\)-dimensional \(M_F\), it has such properties\cite{29–31}:

1. Extract \(x_i\) or \(f_i(x)\) in \(O(\log^2 N)\) time complexity, which are represented with \(O_{M1}\) and \(O_{M2}\):

\[
\begin{align*}
O_{M1}|i\rangle|0\rangle &= |i\rangle|x_i\rangle, \\
O_{M2}|i\rangle|0\rangle &= |i\rangle|f_i(x)\rangle, \\
i &= 0, 1, ..., N - 1.
\end{align*}
\]

2. Extract the norm of part of \(x\) or \(F(x)\) in \(O(\log^2 N)\) time complexity, which are represented with \(O_{M3}\) and \(O_{M4}\):

\[
\begin{align*}
O_{M3}|i\rangle|j\rangle|0\rangle &= |i\rangle|j\rangle|d_{ijx}\rangle, \\
O_{M4}|i\rangle|j\rangle|0\rangle &= |i\rangle|j\rangle|d_{ijy}\rangle, \\
i &= 0, 1, ..., n, \; j &= 0, 1, ..., 2^i - 1.
\end{align*}
\]

3. The time complexity of Updating the whole \(M_F\) is \(O(M \log^2 N)\), where \(M\) is the sum of the number of \(x_i\) and \(f_j(x)\) to be updated.

### B. Preprocessing

Now we introduce three steps of each iteration in QNM. In preprocessing step, we build operations to extract the information of the system of linear equations in this iteration, which is defined in Eq.\((10)\) :

\[
A\Delta x = -|b|,
\]

\(A\) and \(|b\rangle\) are defined as:

\[
A = \frac{F'(x)}{\|F'(x)\|_{\text{max}}},
\]

\[
|b\rangle = \frac{1}{C_b} \sum f_i(x)|i\rangle,
\]

where \(C_b\) is the normalization constant of \(F(x)\), \(\|F'(x)\|_{\text{max}} := \max_{j,k} |F'(x)|_{jk}\). We assume that an upper bound of \(\|F'(x)\|_{\text{max}}\) can be estimated and we regard the upper bound as \(\|F'(x)\|_{\text{max}}\).

We build \(O_b\) to prepare \(|b\rangle\) and \(O_{A1}, O_{A2}\) to extract the information of matrix \(A\). \(O_b\) is defined as:

\[
O_b(0) = |b\rangle.
\]

We use the method proposed in \cite{32} to build \(O_b\). In \(O_b\)'s construction process, we query \(O_{M4} O(\log N)\) times. \(O_{A1}\) and \(O_{A2}\) are defined as:

\[
\begin{align*}
O_{A1}|j, l\rangle &= |j, h(j, l)\rangle, \\
O_{A2}|j, k, z\rangle &= |j, k, z \oplus A_{jk}\rangle,
\end{align*}
\]

where \(h(j, l)\) is the column number of \(A\)'s \(l\)-th non-zero element. It is easy to see that \(O_{A1}\) and \(O_{f1}\) are the same. We only need to build \(O_{A2}\). For simplicity, we let \(|z\rangle = |0\rangle\), \(O_{A2}\) is rewritten as:

\[
O_{A2}|j, k, 0\rangle = |j, k, F'(x)|^{-1}_{\text{max}} \partial f_j(x)\rangle
\]

To build \(O_{A2}\), we need a quantum circuit to compute \(\frac{\partial f_j(x)}{\partial x_k}\). We use finite difference method (FDM) to approximate \(\frac{\partial f_j(x)}{\partial x_k}\). For example, Eq.\((17)\) gives an approximation of \(\frac{\partial f_j(x)}{\partial x_k}\):

\[
\frac{\partial f_j(x)}{\partial x_k} \approx f_j(x + \Delta x_k) - f_j(x) \Delta x_k
\]

For each \(f_j(x)\), we extract \(x(j)\) by querying \(O_{f1}\) and \(O_{M1} d\) times. Next we use \(O_{f2}\) to compute \(|f_j(x)\rangle\). We can also change \(|x(j)\rangle\) to \(|x'(j) + \Delta x_k\rangle\) and compute \(|f_j(x'(j) + \Delta x_k)\rangle\). Finally we compute \(A_{jk}\) and uncompute the ancilla qubits we used. The whole process is shown as follows:

\[
\begin{align*}
&|j, k\rangle|0\rangle|0\rangle \\
&\xrightarrow{O_{f1}, O_{M1}} |j, k, x(j)\rangle|0\rangle \\
&\xrightarrow{O_{f2}, FDM} |j, k, x(j)\rangle|f_j(x)\rangle \\
&\xrightarrow{\text{uncompute}} |j, k\rangle|0\rangle|f_j(x)\rangle \\
&\xrightarrow{A_{jk}} |j, k\rangle|0\rangle
\end{align*}
\]

### C. Processing

In processing step, we use QLSS proposed in \cite{7} to solve Eq.\((10)\). We input \(O_b, O_{A1}\) and \(O_{A2}\) and get \(\Delta x\), the normalized solution of Eq.\((10)\). The success rate of the QLSS is

\[
p = \frac{\|A^{-1}|b\rangle\|^2}{\alpha^2},
\]

\(\alpha\) is defined as:

\[
\alpha := \frac{4}{d} \sum_{j=0}^{2^d - 1} \sum_{i=j+1}^{2^d} \left( \frac{2c}{c + i} \right) \left( \frac{c}{2^c} \right),
\]

where \(j_0 = \sqrt{c \log(4c/\epsilon)}\) and \(c = \kappa^2 \log(\kappa/\epsilon)\). \(p\) can be estimated with quantum amplitude estimation algorithm\cite{33}.

We execute QLSS multiple times and use \(l_{\infty}\) tomography\cite{28} to sample output state \(|\Delta x\rangle\) and get the sampled state \(|\Delta x\rangle\). The \(l_{\infty}\) tomography is shown in theorem 1, the sample times is \(O(\log N)/\epsilon^2\), where \(\epsilon\) represents the sample error. We also compute the normalization constant of \(\Delta x\): \(c_{\Delta x} = \|\Delta x\|_2\). Combine Eq.\((10)\),
Eq. (11), Eq. (12) and Eq. (18), we have
\[ C_{\Delta x} = \frac{\alpha C_b \sqrt{b}}{||F'(\bar{x})||_{max}}. \] (20)

**Theorem 1** (\(l_\infty\) tomography[28]). Given access to unitary \(U\) such that \(U|0\rangle = |x\rangle\) and its controlled version in time \(T(U)\), there is a tomography algorithm with time complexity \(O(T(U) \log N)\) that produces unit vector 
\(\bar{X} \in R^N\) such that \(||\bar{X} - x||_\infty \leq \varepsilon_s\) with probability at least \((1 - 1/poly(N))\).

**D. Postprocessing**

In postprocessing step, we use the sampled state \(|\bar{x}\rangle\) to update \(M_F\) and determine whether to stop the iteration.

We first update the root node of \(M_F\): \(x_i\) and \(f_i(\bar{x})\).

The update of \(x_i\) is:
\[ |x\rangle \rightarrow |x + C_{\Delta x} \bar{x}\rangle \approx |F^{-1}(x)F(\bar{x})\rangle \] (21)

From \(|\bar{x}\rangle\) we compute the changed \(f_i(\bar{x})\) and update the changed \(f_i(x)\). We use \(N_x, N_f\) to represent the number of changed \(x_i\) and \(f_i(x)\) respectively. \(N_x, N_f\) satisfy:
\[ N_x < \frac{\log N}{\epsilon^2}, \quad N_f < \frac{\log N}{\epsilon^2}. \] (22)

Then the update complexity of the whole \(M_F\) is \(O(\frac{\log^2 N}{\epsilon^2})\).

The iteration cutoff condition is \(||F(\bar{x})|| < \varepsilon\). At the end of each iteration, we use \(O_{M4}\) introduced in Eq. (9) to compute \(||F(\bar{x})||\), we set \(i = 0, j = 0\) and have
\[ O_{M4}[0][0][0] = |0\rangle |0\rangle |0\rangle \] (23)

If \(||F(\bar{x})|| < \varepsilon\), we stop iteration, otherwise, we execute the next iteration until the iteration stops.

Up to now, we have introduced the whole process of QNM. The process is also shown in Algorithm 2. The output of QNM is the updated \(M_F\), it saves the solution of the system of nonlinear equations.

**Algorithm 2** Quantum Newton’s method

**Input:** initial \(M_F, O_{f_1}, O_{f_2}\), sampled error \(\varepsilon_s\), and accuracy error \(\varepsilon\).

**Output:** \(M_F\).

1: Construct \(O_b, O_{A1}\) and \(O_{A2}\).
2: Input \(O_b, O_{A1}\) and \(O_{A2}\) into QLSS introduced in Sec. III C, execute QLSS \(O(\frac{\log N}{\epsilon^2})\) times and get \(|\bar{x}\rangle\) with \(l_\infty\) tomography.
3: Compute \(C_{\Delta x}\) with Eq. (20).
4: Update \(M_F\) with the process described in Sec. III D.
5: Compute \(||F(\bar{x})||\), if \(||F(\bar{x})|| \leq \varepsilon\), turn to step 6. Otherwise, turn to step 1.
6: Return \(M_F\).
7: End.

**IV. COMPLEXITY**

In this section, we analyze the query complexity and time complexity of QNM. First, we analyze the query complexity.

**A. Query complexity**

In QNM, we use \(O_{f_1}, O_{f_2}\) and \(M_F\) to build other operations. Now we analyze the query complexity of \(O_{f_1}, O_{f_2}\) and \(M_F\).

In \(O_b\)’s construction process, we query \(M_F\) \(O(\log N)\) times[32]. In \(O_{A2}\)’s construction process introduced in Sec. III B, we query \(O_{f_1}, M_F\) \(O(d)\) times and query \(O_{f_2}\) \(O(1)\) times.

In QLSS, improved by variable-time amplitude amplification[34], the query complexity of \(O_{A1}, O_{A2}\) and \(O_b\) is \(O(\log^2(\log^2 d/\epsilon))\)[7], \(\kappa\) represents the condition number of matrix \(A\). As shown in Theorem 1, the sampling complexity is \(O(\frac{\log N}{\epsilon^2})\). In postprocessing step, we query \(M_F\) \(O(1)\) times and we can omit it. Therefore in each iteration, the query complexity of \(M_F\) is
\[ O((d + \log N)\frac{d\log N}{\epsilon^2} \text{poly}(\log(\log d/\epsilon))), \] (24)

the query complexity of \(O_{f_1}\) is
\[ O(\frac{d^2\kappa\log N}{\epsilon^2} \text{poly}(\log(\log d/\epsilon))), \] (25)

and the query complexity of \(O_{f_2}\) is
\[ O(\frac{d\log N}{\epsilon^2} \text{poly}(\log(\log d/\epsilon))). \] (26)

**B. Time Complexity**

Next, we analyze the time complexity of QNM. Here we regard the time complexity of \(O_{f_1}\) and \(O_{f_2}\) as \(O(1)\). In construction process of \(O_b\), we query \(O_{M4}\) \(\log N\) times and perform \(\log N\) quantum arithmetic operations, the time complexity of quantum arithmetic is \(O(\log(\log 1/\epsilon)))[35]\, therefore the time complexity of \(O_b\) is:
\[ T(O_b) = O(\log N \times (\text{poly}(\log(1/\epsilon)) + \log^2 N)) \] (27)

\(O_{A2}\) is built with \(O_{f_1}, O_{f_2}, O_{M4}\) and quantum arithmetic, the time complexity of \(O_{A2}\) is
\[ T(O_{A2}) = O(d\log^2 N + \text{poly}(\log(1/\epsilon))) \] (28)

Then the time complexity of QLSS is[7]
\[ T(\text{QLSS}) = O(d^2\text{poly}(\log(\log d/\epsilon))(T(O_b) + T(O_{A1}) + T(O_{A2}))) \]
\[ = O(d^2\text{poly}(\log(\log d/\epsilon))(3N + d\log^2 N)) \] (29)
The $\kappa$-dependence of $T(\text{QLSS})$ can be improved from quadratic to nearly linear with variable-time amplitude amplification\[7, 34\], then

$$T(\text{QLSS}) = O(d\kappa \text{poly}(\log(\kappa/\epsilon)(\log^3 N + d\log^2 N)))$$  \hspace{1cm} (30)

In $l_\infty$ tomography, we execute QLSS $O(\log^3 N)$ times. Finally, the time complexity of updating $M_F$ is $O(d\kappa \log^2 N)$. In summary, the time complexity of one iteration is

$$T_q = O\left(\frac{\log^3 N}{\epsilon^2 d\kappa} \text{poly}(\log(\kappa/\epsilon)(\log N + d))\right)$$  \hspace{1cm} (31)

In each iteration of classical Newton’s method, we need to solve a system of linear equations, and one of the best algorithms to solve the system of linear equations is conjugate gradient algorithm\[36\], the complexity of conjugate gradient algorithm is $O(Nd\kappa \log(1/\epsilon))$. The complexity of other processes in classical Newton’s method can be ignored. Therefore the time complexity of one iteration of classical Newton’s method is

$$T_c = O(Nd\kappa \log(1/\epsilon))$$  \hspace{1cm} (32)

Compare Eq.(31) and Eq.(32), we find that the $\kappa$-dependence and $\epsilon$-dependence are nearly the same. The difference is mainly in $N$, $d$ and $\epsilon_s$. In general, most of the real systems are sparse, $d$ has little effect on complexity, we can ignore $d$. The complexity of QNM contains $\epsilon_s$. We do some numerical simulations in Sec.V and find when $\epsilon_s > > 1/\sqrt{N}$, QNM is still effective, then the time complexity of QNM is sublinear with $N$. Therefore, compared with classical Newton’s method, QNM provides a quantum advantage.

V. APPLICATION AND NUMERICAL SIMULATION

QNM can be applied to many fields, including computational fluid dynamics, nonlinear dynamics, economics, etc., to solve nonlinear problems in these fields. A general solution process is divided into the following steps: (1) Given a specific nonlinear problem, such as a nonlinear differential equation; (2) Discretize the problem by numerical methods, such as FDM, FEM, etc., to obtain the system of nonlinear equations to be solved. (3) Use QNM to solve the system of nonlinear equations and obtain the solution of the problem.

Compared with classical Newton’s method, we have an $l_\infty$ tomography in each iteration and update $x$ with sampled state, it will affect iteration speed or precision of QNM. To prove the effectiveness of QNM, we do some numerical simulations to test the influence of sampling error $\epsilon_s$ on QNM.

We choose two problems: (1) Nonlinear first-order diffusion problem; (2) Beam lateral vibration problem. We use FDM to discretize the two problems and use QNM to solve the systems of nonlinear equations obtained by FDM. We first introduce the details of these two problems.

A. Problems and difference format

1. Nonlinear first-order diffusion problem

The first problem is solving a nonlinear first-order diffusion problem with our method, the problem is defined as:

$$\begin{aligned}
\frac{\partial u(x,t)}{\partial t} &= g(u(x,t), \frac{\partial u(x,t)}{\partial x}) + f(x,t), -1 < x < 1, t > 1, \\
g(u(x,t)) &= -1 + x \sin(u(x,t)), \\
f(x,t) &= 2e^{2t}[x^2 - 1 + x - x^2 \sin(e^{2t}e^2 - 1)]
\end{aligned}$$  \hspace{1cm} (33)

The initial condition and boundary condition are:

$$\begin{aligned}
u(x,0) &= x^2 - 1, -1 < x < 1; \\
u(-1,t) &= 0, t > 0.
\end{aligned}$$  \hspace{1cm} (34)

We discrete the equation in a such way:

$$\begin{aligned}
x_i &= -1 + (i + 1)h, i = 0, 1, ..., N_1 - 1, h = \frac{2}{N_1}, \\
t_j &= (j + 1)s, j = 0, 1, ..., N_2 - 1, s = \frac{1}{N_2}.
\end{aligned}$$  \hspace{1cm} (35)

We define $u_{i,j} = u(x_i, t_j)$, $u_{i,j,x}$ and $u_{i,j,t}$ are written as:

$$\begin{aligned}
u_{i,j,x} &= \frac{u_{i,j} - u_{i-1,j}}{h}, \\
u_{i,j,t} &= \frac{u_{i,j} - u_{i,j-1}}{s}.
\end{aligned}$$  \hspace{1cm} (36)

Then we have the following system of nonlinear equations:

$$\begin{aligned}
u_{i,j,t} + (1 - x_i \sin(u_{i,j}))u_{i,j,x} - f(x_i, t_j) &= 0, \\
u_{i,j,1} - u_{i-1,j} &= x_i^2 - 1.
\end{aligned}$$  \hspace{1cm} (37)

The Jacobi Matrix of Eq.(37) is an $N_1 \times N_2$-dimensional sparse square matrix and the position of non-zero element is regular, the position of non-zero element of $m = N_1 \times j + i$ row is represented as:

$$S(m) = \begin{cases}
\{m\}, & i = 0, j = 0, \\
\{m - 1, m\}, & i \neq 0, j = 0, \\
\{m - N_1, m\}, & i = 0, j \neq 0, \\
\{m - N_1, m - 1, m\}, & i \neq 0, j \neq 0.
\end{cases}$$  \hspace{1cm} (38)

The oracles $O_{f_1}$ and $O_{f_2}$ of Eq.(37) can be constructed by realizing Eq.(38) and Eq.(37) with some quantum arithmetic circuits. The gate complexity is $O(\log N + \text{poly}(\log(1/\epsilon))\times)\[37\]. When considering the complexity of $O_{f_1}$ and $O_{f_2}$ in the overall time complexity of our method, we can find that the complexity of $O_{f_1}$ and $O_{f_2}$ can be ignored, and the overall time complexity does not change.
2. Beam lateral vibration problem

The second problem is a beam lateral vibration problem, the problem is defined as:

\[
g(x) \omega_{xx} + 2 g x \omega_{x} + \mu(x) \frac{\omega_{xx}}{\omega_{x}} + f(x, t) = 0, \quad \omega(x, 0) = 0, \quad \omega_t(x, 0) = 0.
\]

We use C++ to implement our numerical simulations. Also be constructed by realizing Eq.(44) and Eq.(43) with nonlinear first-order diffusion problem described in Problem 1, the problem is defined as:

\[
(x, t) = e^{-x^2-t^2}. \quad \text{The initial condition and the boundary condition are:}
\]

\[
\begin{align*}
\omega(x, 0) &= e^{-x^2}, \quad \omega_t(x, 0) = 0, \\
\omega(\pm 4, t) &= 0, \quad \omega_{x}(\pm 4, t) = 0.
\end{align*}
\]

We also discuss the application of QNM do some numerical simulations. In this paper, we developed QNM for solving system of nonlinear equations. Similar to the nonlinear first-order diffusion problem described in Sec. V A 1, the oracles \(O_1 \) and \(O_2 \) of this problem can also be constructed by realizing Eq.(44) and Eq.(43) with some quantum arithmetic circuits.

B. Numerical simulations

Now we show the details of our numerical simulations. We use C++ to implement our numerical simulations. In our simulation, we use the LU decomposition linear solver[38] in the C++ Eigen template library to solve the linear system in each iteration of QNM and get \(\Delta x\). then we execute Algorithm 3 in [28] to get the sampled \(\Delta x\) and update \(x\) with the sampled \(\Delta x\). We set the difference points of the two dimensions as \(N_1 = 200\), \(N_2 = 200\), then we have \(N = 40000\)-dimensional system of nonlinear equations.

The numerical simulations of the two problems are shown in Fig.3. From the numerical simulations, we get such results: (1) When \(\epsilon_s = 1/\sqrt{N} = 0.005\), QNM converges for the two problems. When \(\epsilon_s > 1/\sqrt{N}\), QNM also converges, the upper bound of \(\epsilon_s\) when QNM converges for the two problems is 0.05 and 0.1 respectively. (2) Sample error \(\epsilon_s\) slows down the iteration speed and increases iteration times. Compared with \(N\), iterations times has little influence on the complexity of QNM, so we can ignore the influence on iteration times. (3) When \(\epsilon_s\) is relatively large, \(||F(x)|||\) cannot converge to the result that the sampling error is 0, so QNM can only get an approximate solution.

We also test the effects of \(\epsilon_s\) on convergent \(||F(x)|||\). In Fig.4, we repeat QNM 20 times to get the statistical results of the effects of \(\epsilon_s\) on convergent \(||F(x)|||\). We get the following results: (1) As \(\epsilon_s\) increases, the convergent \(||F(x)|||\) also increases. In Fig.4(a), there is an interval in which \(\epsilon_s\) increases and convergent \(||F(x)|||\) decreases, but the overall trend that convergent \(||F(x)|||\) increases with the increase of \(\epsilon_s\) remains unchanged. (2) In both problems, convergent \(||F(x)|||\) << \(\epsilon_s\), so even if we take a relatively large \(\epsilon_s\), we can still get a result error much lower than \(\epsilon_s\).

From these numerical simulations, we can see that when \(\epsilon_s >> 1/\sqrt{N}\), QNM is still effective, the convergent \(||F(x)|||\) is much smaller than \(\epsilon_s\).

VI. CONCLUSION

In this paper, we developed QNM for solving system of nonlinear equations. We give the details of the QNM, analyze the query complexity and time complexity of QNM. We also discuss the application of QNM do some numerical simulations to study the influence of \(\epsilon_s\) on QNM. We find when \(\epsilon_s >> 1/\sqrt{N}\), QNM is still effective, so the complexity of QNM is sublinear with \(N\). Compared with optimal classical algorithm, QNM provides a quantum advantage. We also notice that when \(\epsilon_s < 1/\sqrt{N}\), the complexity of QNM is superlinear with \(N\), which is worse than classical Newton’s method.

QNM has the potential to provide quantum acceleration in many fields, such as nonlinear finite element analysis[1], nonlinear dynamics[2], nonlinear programming[3] and so on. In the future, we will develop the application of QNM in various practical problems, test the impact of \(\epsilon_s\) on QNM in various specific problems.
FIG. 3. Numerical Results. Panels (a),(b) represent the influence of sample error $\epsilon_s$ on the convergence process of nonlinear first-order diffusion problem and beam lateral vibration problem respectively.

FIG. 4. Relationship between $\epsilon_s$ and convergent $||F(x)||$. Panels (a),(b) represent nonlinear first-order diffusion problem and beam lateral vibration problem respectively.

ACKNOWLEDGEMENT

This work was supported by the National Key Research and Development Program of China (Grant No. 2016YFA0301700), the National Natural Science Foundation of China (Grants Nos. 11625419), the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDB24030600), and the Anhui Initiative in Quantum Information Technologies (Grants No. AHY080000).

[1] J. N. Reddy, *Nonlinear finite element analysis* (Oxford University Press, New York, 2004).
[2] J. M. T. Thompson and H. B. Stewart, *Nonlinear dynamics and chaos* (John Wiley & Sons, 2002).
[3] O. L. Mangasarian, *Nonlinear programming* (SIAM, 1994).
[4] W. C. Rheinbolt, *Methods for Solving Systems of Nonlinear Equations* (Society for Industrial and Applied Mathematics, 1974).
[5] A. Burden, R. Burden, and J. Faires, *Numerical Analysis, 10th ed.* (2016).
[6] A. W. Harrow, A. Hassidim, and S. Lloyd, Physical Review Letters 103, 150502 (2009).
[7] A. M. Childs, R. Kothari, and R. D. Somma, SIAM Journal on Computing 46, 1920 (2017), publisher: Society for Industrial and Applied Mathematics.
[8] Y. Subaši, R. D. Somma, and D. Orsucci, *Physical Review Letters* **122** (2019), 10.1103/physrevlett.122.060504.

[9] B. D. Clader, B. C. Jacobs, and C. R. Sprouse, *Physical Review Letters* **110** (2013), 10.1103/physrevlett.110.250504.

[10] D. W. Berry, *Journal of Physics A: Mathematical and Theoretical* **47**, 105301 (2014).

[11] A. Montanaro and S. Pallister, *Physical Review A* **93** (2016), 10.1103/physreva.93.032324.

[12] D. W. Berry, A. M. Childs, A. Ostrander, and G. Wang, *Communications in Mathematical Physics* **356**, 1057–1081 (2017).

[13] T. Xin, S. Wei, J. Cui, J. Xiao, I. Arrazola, L. Lamata, X. Kong, D. Lu, E. Solano, and G. Long, *Physical Review A* **101** (2020), 10.1103/physreva.101.032307.

[14] Y. Cao, A. Papageorgiou, I. Petras, J. Traub, and S. Kais, *New Journal of Physics* **15**, 013021 (2013).

[15] P. C. S. Costa, S. Jordan, and A. Ostrander, *Physical Review A* **99** (2019), 10.1103/physreva.99.012323.

[16] F. Fillion-Gourdeau, S. MacLean, and R. Laflamme, *Physical Review A* **95** (2017), 10.1103/physreva.95.042343.

[17] A. Engel, G. Smith, and S. E. Parker, *Physical Review A* **100** (2019), 10.1103/physreva.100.062315.

[18] J. M. Arrazola, T. Kulajdzievi, C. Weedbrook, and S. Lloyd, *Physical Review A* **100** (2019), 10.1103/physreva.100.032306.

[19] N. Linden, A. Montanaro, and C. Shao, “Quantum vs. classical algorithms for solving the heat equation,” (2020), arXiv:2004.06516 [quant-ph].

[20] A. M. Childs and J.-P. Liu, *Communications in Mathematical Physics* **375**, 1427–1457 (2020).

[21] A. M. Childs, J.-P. Liu, and A. Ostrander, “High-precision quantum algorithms for partial differential equations,” (2020), arXiv:2002.07868 [quant-ph].

[22] S. K. Leyton and T. J. Osborne, “A quantum algorithm to solve nonlinear differential equations,” (2008), arXiv:0812.4423 [quant-ph].

[23] M. Lubasch, J. Joo, P. Moinier, M. Küffner, and D. Jaksch, *Physical Review A* **101**, 010301 (2020), publisher: American Physical Society.

[24] J.-P. Liu, H. Oie Kolden, H. K. Krovi, N. F. Loureiro, K. Trivisa, and A. M. Childs, “Efficient quantum algorithm for dissipative nonlinear differential equations,” (2020), arXiv:2011.03185 [quant-ph].

[25] S. Lloyd, G. D. Palma, C. Gokler, B. Kiani, Z.-W. Liu, M. Marvian, F. Tennie, and T. Palmer, “Quantum algorithm for nonlinear differential equations,” (2020), arXiv:2011.06571 [quant-ph].

[26] P. Qian, W.-C. Huang, and G.-L. Long, arXiv:1903.05608 [quant-ph] (2019), arXiv:1903.05608.

[27] P. Rebentrost, M. Schuld, L. Wossnig, F. Petruccione, and S. Lloyd, *New Journal of Physics* **21**, 073023 (2019).

[28] I. Kerenidis, J. Landman, and A. Prakash, “Quantum algorithms for deep convolutional neural networks,” (2019), arXiv:1911.01117 [quant-ph].

[29] I. Kerenidis and A. Prakash, (2016).

[30] V. Giovannetti, S. Lloyd, and L. Maccone, *Physical Review Letters* **100** (2008), 10.1103/physrevlett.100.160501.

[31] V. Giovannetti, S. Lloyd, and L. Maccone, *Physical Review A* **78** (2008), 10.1103/physreva.78.052310.

[32] L. Grover and T. Rudolph, arXiv:quant-ph/0208112 (2002), arXiv:quant-ph/0208112.

[33] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, “Quantum amplitude amplification and estimation,” (2000), arXiv:quant-ph/0005055 [quant-ph].

[34] A. Ambainis, “Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations,” (2010), arXiv:1010.4458 [quant-ph].

[35] K. Mitarai, M. Kitagawa, and K. Fujii, *Physical Review A* **99**, 012301 (2019).

[36] J. Shewchuk, Online (1994).

[37] M. A. Nielsen and I. Chuang, *American Journal of Physics* **70**, 558 (2002).

[38] Y. Saad, *Iterative methods for sparse linear systems* (SIAM, 2003).