Quantum Algorithm to Cubic Spline Interpolation

Changpeng Shao
Academy of Mathematics and Systems Science,
Chinese Academy of Sciences
Beijing 100190, China
(Dated: April 3, 2018)

In this work, we provide another application of HHL algorithm in cubic spline interpolation problem. In this problem, the condition number is small, the Hamiltonian simulation is efficient and quantum state preparation is efficient. So the quantum algorithm obtained by HHL algorithm toward this problem actually achieves an exponential speedup with no restrictions. This will be another application of HHL algorithm besides the work of Clader et al. [1] to the electromagnetic scattering cross-section problem with no caveats.

I. INTRODUCTION

HHL algorithm [2] is an important quantum linear algebra based subroutine of many quantum algorithms to machine learning problems (for example, see applications in [3], [4], [5]). However, because of its caveats in quantum state preparation, Hamiltonian simulation and the dependence on condition number, many recently discovered quantum machine learning algorithms also possess several of the restrictions. On one hand, finding more applications of HHL algorithm is an important task that can provide more problems that quantum computer can speedup. Since under certain conditions, such quantum algorithms will achieve exponential speedup. And on the other hand, finding more applications of HHL algorithm with few restrictions also deserve of studying. Since, such problems will achieve exponential speedup with no conditions. One such application of HHL algorithm seems to belong to Clader et al. [1] at 2013 of studying electromagnetic scattering cross-section problem via finite element method.

A typical application of HHL algorithm is linear regression [6], [7], [8], since the quantum state of the solution is enough to do the prediction on new data. And the restrictions related to this problem will be the Hamiltonian simulation and condition number. From the point of practicality, locally weighted linear regression is more useful. It is easier than high degree regression. Also when considering about Hamiltonian simulation, locally weighted linear regression is more suitable to study by quantum computer [9], since it corresponds to low rank matrix, whose Hamiltonian simulation can be implemented efficiently [5], [10]. In this problem, the only caveat will be the condition number, which are often hard to estimate.

A similar research topic closely related to linear regression will be polynomial interpolation or approximation [11]. We have global interpolation method like Lagrange interpolation, Newton interpolation and Hermite interpolation. Locally, we can apply piecewise linear interpolation, cubic Hermite interpolation and cubic spline interpolation, among which cubic spline interpolation performs quite well than others. Cubic spline interpolation is smoother than cubic Hermite interpolation, also it can avoid Runge’s phenomenon. It contains a strong convergence and stability property, which is useful in theory and in practice. More importantly (from the point of quantum computer), it corresponds to a linear system, whose coefficient matrix is diagonally dominant and tridiagonal. And the condition number is bounded by a small constant. This means HHL algorithm can play an important role in this problem with no caveats. When obtaining the quantum state of the solution by HHL algorithm, the evaluation on new data can also solved easily by swap test. At this case, the quantum state of the new data only contains two nonzero entries, which implies that the quantum state of the new data can be prepared efficiently. Therefore, cubic spline interpolation seems to be a very good application of HHL algorithm. In this work, we will focus on the analysis of HHL algorithm on this problem.

The structure of this work is as follows: In section II, we present some preliminaries about cubic spline interpolation. In section III, we analyze the condition number of the linear system in cubic spline interpolation. Finally, in section IV, we apply HHL algorithm to solve the cubic spline interpolation problem.

II. PRELIMINARIES OF CUBIC SPLINE INTERPOLATION

In this section, we briefly review the cubic spline interpolation method. Since the aim of this work is providing an new application of HHL algorithm, so we will not go deeper in cubic spline interpolation and its applications, generalizations. Given a data set of \( n + 1 \) samples

\[
\mathcal{X} = \{(x_i, y_i) : i = 0, 1, \ldots, n \text{ and } x_i \neq x_j \text{ if } i \neq j\},
\]

where \( x_i, y_i \in \mathbb{R} \). We also assume that \( a = x_0 < x_1 < \cdots < x_n = b \). The spline function \( S(x) \) is a function satisfying (see figure 1 as an intuitive example):

1. \( S(x) \) is second differentiable in interval \([a, b] \);

\[ \]
2. \( S(x) \) is a polynomial of degree 3 in each subinterval \([x_i, x_{i+1}]\) for all \( i = 0, 1, \ldots, n-1 \);

3. \( S(x_i) = y_i \) for all \( i = 0, 1, \ldots, n \).

The spline function \( S(x) \) in this type is called periodic splines.

There are several methods that can be used to find the spline function \( S(x) \) according to its corresponding conditions \([11], [12]\). The main idea are the same. In the following, we follow the idea of \([13]\) by considering the second derivatives \( S''(x_i) = M_i \) (\( i = 0, 1, \ldots, n \)). In the following, we will focus on finding all \( M_i \). By Lagrange interpolation, with the boundary condition \( C''(x_i) = M_i \) and \( C''(x_{i+1}) = M_{i+1} \), we can interpolate each \( C'' \) on interval \([x_i, x_{i+1}]\) in the following form

\[
C''(x) = M_i \frac{x_{i+1} - x}{h_i} + M_{i+1} \frac{x - x_i}{h_i},
\]

where

\[
h_i = x_{i+1} - x_i.
\]

Integrating the equation \((5)\) twice and using the conditions \( C_i(x_i) = y_i \) and \( C_i(x_{i+1}) = y_{i+1} \), we have

\[
C_i(x) = \frac{M_i}{6h_i} (x_{i+1} - x)^3 + \frac{M_{i+1}}{6h_i} (x - x_i)^3 + \left( y_i - \frac{M_i h_i^2}{6} \right) \frac{x_{i+1} - x}{h_i} + \left( y_{i+1} - \frac{M_{i+1} h_{i+1}^2}{6} \right) \frac{x - x_i}{h_i}.
\]

Therefore,

\[
C_i'(x_{i+1}) = \frac{(M_i + 2M_{i+1}) h_i}{6} + \frac{y_{i+1} - y_i}{h_i},
\]

\[
C_{i+1}'(x_{i+1}) = -\frac{(2M_{i+1} + M_{i+2}) h_{i+1}}{6} + \frac{y_{i+2} - y_{i+1}}{h_{i+1}}.
\]

The two values should equal to each other because of the second equality in formula \((1)\), so

\[
\mu_{i+1} M_i + 2M_{i+1} + \lambda_{i+1} M_{i+2} = d_{i+1},
\]

where

\[
\mu_{i+1} = \frac{h_i}{h_i + h_{i+1}},
\]

\[
\lambda_{i+1} = 1 - \mu_{i+1} = \frac{h_{i+1}}{h_i + h_{i+1}},
\]

\[
d_{i+1} = 6S\{x_i, x_{i+1}, x_{i+2}\},
\]

\( i = 0, 1, \ldots, n-2 \),

and \( S\{x_i, x_{i+1}, x_{i+2}\} \) is the Newton divided difference. It is defined recursively,

\[
S\{x_i, x_{i+1}, x_{i+2}\} = S\{x_{i+1}, x_{i+2}\} - S\{x_i, x_{i+2}\} \frac{x_{i+2} - x_i}{x_{i+2}},
\]

\[
S\{x_i, x_{i+1}\} = S\{x_{i+1}\} - S\{x_i\} \frac{x_{i+1} - x_i}{x_{i+1} - x_i},
\]

FIG. 1. Spline function: it is a degree 3 polynomial in each interval, also contain certain smooth properties at the joint points.

Because of condition 2, we denote the cubic polynomial in subinterval \([x_i, x_{i+1}]\) as \( C_i(x) \). Then there are totally 4\( n \) unknown parameters we should determine in \( S(x) \). By condition 1 and 3, we have the following 4\( n-2 \) conditions:

\[
\begin{aligned}
C_i(x_i) &= y_i, \\
C_i'(x_{i+1}) &= C_{i+1}'(x_{i+1}), \\
C_i''(x_{i+1}) &= C_{i+1}''(x_{i+1}).
\end{aligned}
\]

We usually add two extra boundary conditions to make the spline function unique. Generally, there are three types of frequently used boundary conditions:

**Type 1:** First derivatives of \( S(x) \) at the endpoints are known:

\[
C'_0(x_0) = f'_0 \quad \text{and} \quad C'_{n-1}(x_n) = f'_n.
\]

The special case \( C'_0(x_0) = C'_{n-1}(x_n) = 0 \) will be called clamped boundary conditions.

**Type 2:** Second derivatives of \( S(x) \) at the endpoints are known:

\[
C''_0(x_0) = f''_0 \quad \text{and} \quad C''_{n-1}(x_n) = f''_n.
\]

The special case \( C''_0(x_0) = C''_{n-1}(x_n) = 0 \) will be called natural boundary conditions.

**Type 3:** Usually, cubic spline interpolation can be used to approximate a given function \( f(x) \), and the data just given as \( y_i = f(x_i) \). When the exact function \( f(x) \) is a periodic function with period \( x_n - x_0 \), we also need \( S(x) \) to be a periodic function with period \( x_n - x_0 \). Thus the condition will be

\[
\begin{aligned}
C_0(x_0) &= C_{n-1}(x_n), \\
C'_0(x_0) &= C'_{n-1}(x_n), \\
C''_0(x_0) &= C''_{n-1}(x_n).
\end{aligned}
\]
with the given information that \( S(x_i) = y_i \).

For type 1 boundary condition, we will have
\[
2M_0 + M_1 = \frac{6}{h_0} (S[x_0, x_1] - f_0'),
\]
\[
M_{n-1} + 2M_n = \frac{6}{h_{n-1}} (f_n' - S[x_{n-1}, x_n]).
\]
Therefore, \( \lambda_0 = \mu_n = 1, d_0 = 6(S[x_0, x_1] - f_0')/h_0 \) and \( d_n = 6(f_n' - S[x_{n-1}, x_n])/h_{n-1}. \) Finally, the system of equations that we need to solve is
\[
\begin{bmatrix}
2 & \lambda_0 \\
\mu_1 & 2 & \lambda_1 \\
\vdots & \ddots & \ddots \\
\mu_{n-1} & 2 & \lambda_{n-1} \\
\lambda_n & \mu_n & 2
\end{bmatrix}
\begin{bmatrix}
M_0 \\
M_1 \\
\vdots \\
M_{n-1} \\
M_n
\end{bmatrix}
= \begin{bmatrix}
d_0 \\
d_1 \\
\vdots \\
d_{n-1} \\
d_n
\end{bmatrix}.
\]

For type 2 boundary condition, we have \( M_0 = f''_0 \) and \( M_n = f''_n \), so we can set \( \lambda_0 = \mu_n = 0 \) and \( d_0 = 2f''_0, d_n = 2f''_n \). Then we need to solve a linear system in the same form as above.

For type 3 boundary condition, we have
\[
M_0 = M_n, \quad \lambda_n M_1 + \mu_n M_{n-1} + 2M_n = d_n,
\]
where
\[
\lambda_n = \frac{h_0}{h_{n-1} + h_0},
\]
\[
\mu_n = 1 - \lambda_n = \frac{h_{n-1}}{h_{n-1} + h_0};
\]
\[
d_n = 6(S[x_0, x_1] - S[x_{n-1}, x_n])/h_0 + h_{n-1}.
\]
The linear system that we need to solve is
\[
\begin{bmatrix}
2 & \lambda_1 \\
\mu_2 & 2 & \lambda_2 \\
\vdots & \ddots & \ddots \\
\mu_{n-1} & 2 & \lambda_{n-1} \\
\lambda_n & \mu_n & 2
\end{bmatrix}
\begin{bmatrix}
M_1 \\
M_2 \\
\vdots \\
M_{n-1} \\
M_n
\end{bmatrix}
= \begin{bmatrix}
d_1 \\
d_2 \\
\vdots \\
d_{n-1} \\
d_n
\end{bmatrix}.
\]

The linear system in (12), (15) are tridiagonal linear system whose coefficient matrices are diagonally dominant. Such linear systems are very stable and has a unique solution. The classical algorithm, such as Gaussian elimination or the chasing method, to solve such linear systems is also not difficult. The complexity is \( O(n) \). In this special case, we will believe that quantum computer can achieve exponential speedup by HHL algorithm.

### III. Bounds on singular values

The Gershgorin type of circle theorem also holds for singular values \( \lambda_i, \mu_i \). Let \( A = (a_{ij})_{n \times n} \) be any complex matrix, denote
\[
r_i = \sum_{1 \leq j \leq n} |a_{ij}|, \quad c_j = \sum_{1 \leq i \leq n} |a_{ij}|, \quad s_i = \max(r_i, c_i).
\]
Then all the singular values of \( A \) lie in
\[
\bigcup_{i=1}^{n} [\max(0, |a_{ii} - s_i)|, a_{ii} + s_i].
\]

Such an estimation may fail to estimate the condition number of matrix \( A \) when the low bound of the interval is zero.

In the linear system (12) and (15), since \( \lambda_i + \mu_i = 1 \) and \( \lambda_i, \mu_i \geq 0 \), by (16), all the singular values of the coefficient matrix of the linear system (12) and (15) lie in
\[
\bigcup_{i=1}^{n} [2 - s_i, 2 + s_i]
\]
In type 1, \( \lambda_0 = \mu_n = 1 > \lambda_i, \mu_i > 0 \) \( (i = 1, 2, \ldots, n - 1) \), then
\[
\max(s_i) \leq \max_{1 \leq i \leq n-3} \{1 + \mu_2, \lambda_i + \mu_i+2, 1 + \lambda_n-2\}.
\]
In type 2, \( \lambda_0 = \mu_n = 0 \) and \( 1 > \lambda_i, \mu_i > 0 \) \( (i = 1, 2, \ldots, n - 1) \), then
\[
\max(s_i) \leq \max_{1 \leq i \leq n-3} \{1, \lambda_i + \mu_i+2\}.
\]
In type 3, \( \lambda_0 = \mu_n = 0 \) and \( 1 > \lambda_i, \mu_i > 0 \) \( (i = 1, 2, \ldots, n - 1) \), then
\[
\max(s_i) \leq \max_{1 \leq i \leq n-2} \{1, \lambda_i + \mu_i+2, \lambda_n-1 + \mu_1, \lambda_n + \mu_2\}.
\]
The right side of (18), (19), (20) will be denoted as \( s \). In each case, \( s < 2 \). Then the condition numbers of the coefficient matrix of (12) and (15) are bounded by \( 4/(2 - s) \). Note that
\[
\lambda_i = \frac{h_i}{h_{i-1} + h_i} \quad \text{and} \quad \mu_i = \frac{h_{i-1}}{h_{i-1} + h_i}.
\]
While \( h_i \) refers to the length of the \( i \)-th interval. We often do not set the length too small, so the upper bound \( s \) of (18), (19), (20) are often smaller than 2 with a reasonable distance. In fact, whatever the value of \( h_i \geq 0 \) is, the condition number are all bounded by 4.

As we can see above, the estimating result is not good enough if we directly apply (16) to estimate the singular values of coefficient matrix of the linear system (12) and (15). Consider the linear system in (12), denote
\[
D = \text{diag}\{h_0, h_0 + h_1, \ldots, h_{n-2} + h_{n-1}, h_{n-1}\}
\]
as a diagonal matrix and the coefficient matrix of (12) as \( A \). Then we multiply \( D \) into the linear system (12).
Actually, this can be done when we construct the linear system \([12]\). Now set \(B = DA\). It has the following matrix form

\[
\begin{bmatrix}
2h_0 & h_0 \\
h_0 & 2(h_0 + h_1) & h_1 \\
& h_1 & 2(h_1 + h_2) & h_2 \\
& & h_2 & 2(h_2 + h_3) \\
& & & \ddots & \ddots & \ddots \\
& & & & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\
& & & & & h_{n-1} & 2h_{n-1}
\end{bmatrix}
\]

We found that \(B\) is symmetry. Now we apply \([16]\) to estimate the singular values of \(B\), they lie in the interval

\[
[0, 3h_{0}] \cup [h_{0} + h_{1}, 3(h_{0} + h_{1})] \cup \cdots \cup [h_{n-2} + h_{n-1}, 3(h_{n-2} + h_{n-1})] \cup [h_{n-1}, 3h_{n-1}].
\]

Then any singular value \(\sigma\) of \(B\) satisfies

\[
\begin{align*}
\min\{h_{0}, h_{0} + h_{1}, & \ldots, h_{n-2} + h_{n-1}, h_{n-1}\} \leq \sigma \\
\leq 3 \max\{h_{0}, h_{0} + h_{1}, & \ldots, h_{n-2} + h_{n-1}, h_{n-1}\}.
\end{align*}
\]

For practical problem, the condition number of \(B\) can be estimated efficiently, and it should be small. Since the error of the cubic spline approximation is controlled by some power of the length of intervals. This means \(h_{i}\) will be small if we want the approximation error is small. However, they cannot too small, otherwise it will bring other troubles of interpolation polynomial. The other two types of cubic spline interpolation can also analyzed similarly.

IV. QUANTUM CUBIC SPLINE INTERPOLATION

For the linear system \([12]\), the condition number is not too large. Also the coefficient matrix is sparse. The classical algorithm to solve this linear system is takes time \(O(n)\). However, by HHL algorithm, this linear system can be solved in time \(O((\log n)/\epsilon)\). And we will get a quantum state of the solution

\[
|M\rangle \propto \sum_{i=0}^{n} M_{i}|i\rangle.
\]

Just like linear regression, one important task of polynomial interpolation is to do further prediction on the new data. Now suppose we are giving a new data \(\tilde{x}\). Assume that \(\tilde{x} \in [x_{i}, x_{i+1}]\), then \(S(\tilde{x}) = C_{i}(\tilde{x})\). By formula \([7]\), we have

\[
C_{i}(\tilde{x}) = M_{i}\left[\frac{(x_{i+1} - \tilde{x})^{3}}{6h_{i}} - \frac{h_{i}(x_{i+1} - \tilde{x})}{6}\right]
+ M_{i+1}\left[\frac{(\tilde{x} - x_{i})^{3}}{6h_{i}} - \frac{h_{i}(\tilde{x} - x_{i})}{6}\right]
+ y_{i}\frac{x_{i+1} - \tilde{x}}{h_{i}} + y_{i+1}\frac{\tilde{x} - x_{i}}{h_{i}}
\]

\[
\triangleq M_{i}X_{i} + M_{i+1}X_{i+1} + Y_{i}.
\]

Then we just need to prepare the quantum state

\[
|X\rangle = X_{i}|i\rangle + X_{i+1}|i+1\rangle.
\]

Certainly, this quantum state can be obtained efficiently. By swap test \([15]\), we can evaluate the inner product of \(|M\rangle\) and \(|X\rangle\), and so evaluate \(S(\tilde{x})\) efficiently in time \(O((\log n)/\epsilon^{2})\) to accuracy \(\epsilon\). Or on the other hand, we can just apply swap test to evaluate \(M_{i}, M_{i+1}\), and \(M_{i+2}\), then according to formula \([9]\) to find out the missed normalization factor. Within the same complexity, we can also evaluate \(S'(\tilde{x})\). Also, we can compute \(S'(\tilde{x})\) and \(S''(\tilde{x})\) within the same time.

Acknowledgements. This work was supported by the NSFC Project 11671388, the CAS Frontier Key Project QYZDJ-SSW-SYS022, China Postdoctoral Science Foundation and the Guozhi Xu Postdoctoral Research Foundation.

[1] B. D. Clader and a. C. R. S. B. C. Jacobs, Phys. Rev. Lett. 110, 250504 (2013).
[2] A. W. Harrow, A. Hassidim, and S. Lloyd, Physical review letters 103, 150502 (2009).
[3] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Nature 549, 195 (2017).
[4] V. Dunjko and H. J. Briegel, arXiv:1709.02779v1 (2017).
[5] P. Rebentrost, A. Steffens, and S. Lloyd, Physical review A 97, 012327 (2018).
[6] M. Schulz, I. Sinayskiy, and F. Petruccione, Phys. Rev. A 94, 022342 (2016).
[7] G. M. Wang, Physical review A 96, 012335 (2017).
[8] N. Wiebe, D. Braun, and S. Lloyd, Physical review letters 109, 050505 (2012).
[9] I. Kerenidis and A. Prakash, arXiv:1704.04992v3 (2017).
[10] S. Lloyd, M. Mohseni, and P. Rebentrost, arXiv:1307.0411 (2013).
[11] R. L. Burden and J. D. Faires, Numerical Analysis, 9th ed. (Brooks Cole, 2011).
[12] T. Sauer, Numerical Analysis, 2nd ed. (Pearson Education, 2012).
[13] Q. Li, N. Wang, and D. Yi, Numerical Analysis (In Chinese) (Tsinghua University Press, 2008).
[14] L. Q. Qi, Linear Algebra and Its Applications 56, 105 (1984).
[15] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, Physical review letters 87, 167902 (2001).