HHL - Algorithmus

Alfred Nguyen

Fakultät der Informatik
Technische Universität München
85758 Garching, Bavaria

June 2023

Gliederung

Gliederung

Einführung

Wir haben schon viel über die wichtigsten Algorithmen gehört

- Shors-Algorithmus
- Grover-Algorithmus

Einführung

Wir haben schon viel über die wichtigsten Algorithmen gehört

- Shors-Algorithmus
- Grover-Algorithmus

Der HHL-Algorithmus

- erstellt von Aram Harrow, Avinatan Hassidim und Seth Lloyd
- lösen von sehr großen linearen Gleichungen

$$A\vec{x} = \vec{b}$$

Motivation

Es löst grundlegendes Probleme in der Mathematik

- Least square fitting
- Optimierungs Probleme
- Simulationen und Imageprocessing
- ...

Das Problem

Gegeben:

- Matrix A der Form $n \times n$
- Vektor $\vec{b}$

Das Problem

Gegeben:

- Matrix A der Form $n \times n$
- Vektor $\vec{b}$

Löse das System:

$$A\vec{x} = \vec{b}$$

$$\vec{x} = A^{-1}\vec{b}$$

Das Problem

Gegeben:

- Matrix A der Form $n \times n$
- Vektor $\vec{b}$

Löse das System:

$$A\vec{x} = \vec{b}$$

$$\vec{x} = A^{-1}\vec{b}$$

Wir sind also daran interessiert das Inverse A^{-1} zu finden

Gliederung

Übersicht

Unser Ziel:

$$|x\rangle = A^{-1}|b\rangle$$

Übersicht

Unser Ziel:

$$|x\rangle = A^{-1}|b\rangle$$

Wenn $A = \overline{A^T} = A^\dagger$:

$$|x\rangle = e^{-iAt}|b\rangle$$

Übersicht

Unser Ziel:

$$|x\rangle = A^{-1}\,|b\rangle$$

Wenn $A = \overline{A^T} = A^\dagger$:

$$|x\rangle = e^{-iAt}\,|b\rangle$$

Wenn $A \neq A^\dagger$:

$$A\vec{x} = \vec{b}$$

$$\begin{pmatrix} 0 & A \\ \overline{A^T} & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \vec{x} \end{pmatrix} = \begin{pmatrix} \vec{b} \\ 0 \end{pmatrix}$$

Übersicht

Unser Ziel:

$$|x\rangle = A^{-1}|b\rangle$$

Wenn $A = \overline{A^T} = A^\dagger$:

$$|x\rangle = e^{-iAt}|b\rangle$$

Wenn $A \neq A^\dagger$:

$$A\vec{x} = \vec{b}$$

$$\begin{pmatrix} 0 & A \\ \overline{A^T} & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \vec{x} \end{pmatrix} = \begin{pmatrix} \vec{b} \\ 0 \end{pmatrix}$$

Wir können A diagonalisieren

$$A = U \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & \lambda_n \end{pmatrix} U^T$$

$$\Rightarrow A^{-1} = U^T \begin{pmatrix} \lambda_1^{-1} & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & \lambda_n^{-1} \end{pmatrix} U$$

Der Algorithmus

Ablauf

1. State Preparation

Der Algorithmus

Ablauf

1. State Preparation
   - Enkodiert Vektor und Matrix in Quanten Computer

Der Algorithmus

Ablauf

1. State Preparation
   - Enkodiert Vektor und Matrix in Quanten Computer
2. Quantum Phase Estimation

Der Algorithmus

Ablauf

1. State Preparation
   - Enkodiert Vektor und Matrix in Quanten Computer
2. Quantum Phase Estimation
   - ermittelt Eigenwerte

Der Algorithmus

Ablauf

1. State Preparation
   - Enkodiert Vektor und Matrix in Quanten Computer
2. Quantum Phase Estimation
   - ermittelt Eigenwerte
3. Ancilla Bit Rotation

Der Algorithmus

Ablauf

1. State Preparation
   - Enkodiert Vektor und Matrix in Quanten Computer
2. Quantum Phase Estimation
   - ermittelt Eigenwerte
3. Ancilla Bit Rotation
   - Invertiert Eigenwerte

Der Algorithmus

Ablauf

1. State Preparation
   - Enkodiert Vektor und Matrix in Quanten Computer
2. Quantum Phase Estimation
   - ermittelt Eigenwerte
3. Ancilla Bit Rotation
   - Invertiert Eigenwerte
4. Inverse Quantum Phase Estimation

Der Algorithmus

Ablauf

1. State Preparation
   - Enkodiert Vektor und Matrix in Quanten Computer
2. Quantum Phase Estimation
   - ermittelt Eigenwerte
3. Ancilla Bit Rotation
   - Invertiert Eigenwerte
4. Inverse Quantum Phase Estimation
   - löst verschränkte Qubits auf

Der Algorithmus

Ablauf

1. State Preparation
   - Enkodiert Vektor und Matrix in Quanten Computer
2. Quantum Phase Estimation
   - ermittelt Eigenwerte
3. Ancilla Bit Rotation
   - Invertiert Eigenwerte
4. Inverse Quantum Phase Estimation
   - löst verschränkte Qubits auf
5. Messung

Der Algorithmus

Ablauf

1. State Preparation
   - Enkodiert Vektor und Matrix in Quanten Computer
2. Quantum Phase Estimation
   - ermittelt Eigenwerte
3. Ancilla Bit Rotation
   - Invertiert Eigenwerte
4. Inverse Quantum Phase Estimation
   - löst verschränkte Qubits auf
5. Messung
   - liest das Ergebnis $|x\rangle$ aus

Gliederung

Einfaches Beispiel

Matrix A und Vektor $\vec{b}$:

$$A = \begin{pmatrix} 1 & -\frac{1}{3} \\ -\frac{1}{3} & 1 \end{pmatrix} \qquad \vec{b} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Einfaches Beispiel

Matrix A und Vektor $\vec{b}$:

$$A = \begin{pmatrix} 1 & -\frac{1}{3} \\ -\frac{1}{3} & 1 \end{pmatrix}$$

$$\vec{b} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Klassische Lösung

$$\vec{x} = \begin{pmatrix} \frac{3}{8} \\ \frac{9}{8} \end{pmatrix}$$

Wahrscheinlichkeits Verhältnis der Lösung:

$$\frac{|x_0|^2}{|x_1|^2} = \frac{\frac{9}{64}}{\frac{81}{64}} = \frac{1}{9}$$

Einfach Beispiel

Eigenvektoren von A sind:

$$\vec{u_0} = \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} \qquad \vec{u_1} = \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

Einfach Beispiel

Eigenvektoren von A sind:

$$\vec{u_0} = \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} \qquad \vec{u_1} = \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

Enkodiert

$$|u_0\rangle = \frac{-1}{\sqrt{2}} |0\rangle + \frac{-1}{\sqrt{2}} |1\rangle \qquad |u_1\rangle = \frac{-1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle$$

Einfach Beispiel

Eigenvektoren von A sind:

$$\lambda_0 = \frac{2}{3} \qquad \lambda_1 = \frac{4}{3}$$

Einfach Beispiel

Eigenvektoren von A sind:

$$\lambda_0 = \frac{2}{3}$$

$$\lambda_1 = \frac{4}{3}$$

Einkodiert:

$$\widetilde{\lambda_0} = 1$$

$$|\widetilde{\lambda_0}\rangle = |01\rangle$$

$$\widetilde{\lambda_1} = 2$$

$$|\widetilde{\lambda_1}\rangle = |10\rangle$$

Einfach Beispiel

Eigenvektoren von A sind:

$$\lambda_0 = \frac{2}{3} \qquad \lambda_1 = \frac{4}{3}$$

Einkodiert:

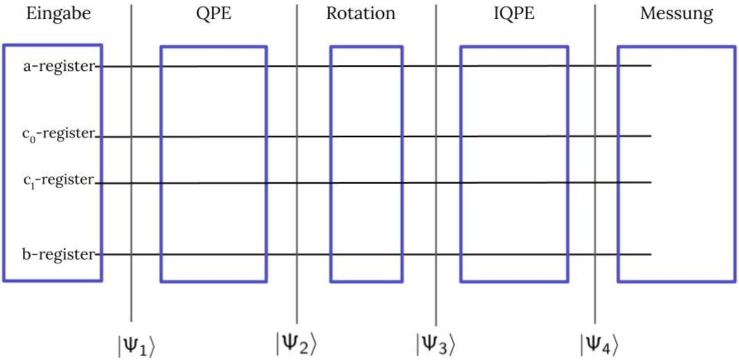
$$\widetilde{\lambda_0} = 1 \qquad \widetilde{\lambda_1} = 2$$

$$|\widetilde{\lambda_0}\rangle = |01\rangle \qquad |\widetilde{\lambda_1}\rangle = |10\rangle$$

Enkodierungsschema:

$$\widetilde{\lambda}_j = N\lambda_j t/2\pi$$

Ablauf/Quantum Circuit



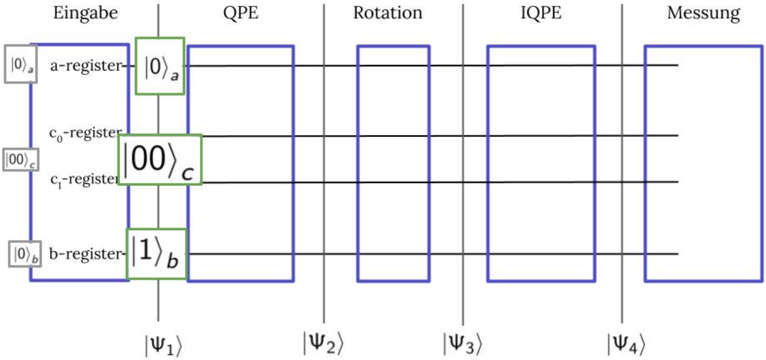
1. Anzahl Qubit for a-register: 1
2. Anzahl Qubits für das c-Register: N = 2
3. Anzahl Qubits für $\vec{b}$: $n_b = log_2(N) = log_2(2) = 1$

State Preparation

- $\vec{b}$ wird als Quantenzustand $|b\rangle$ kodiert
- in unserem Fall ist es sehr einfach

$$\vec{b} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Leftrightarrow |b\rangle = 0\,|0\rangle + 1\,|1\rangle = |1\rangle$$

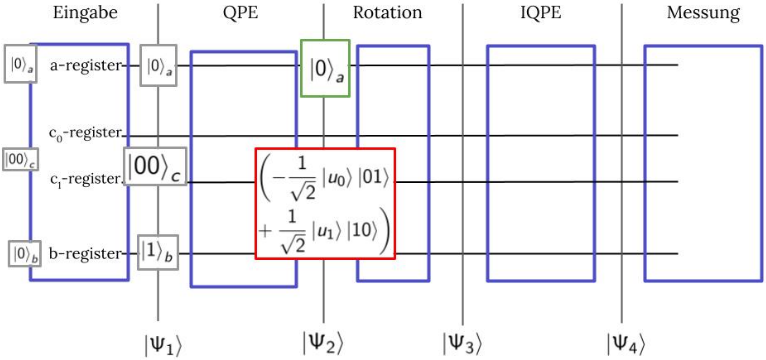
State Preparation



Wir starten im 1 Zustand

$$|\Psi_1\rangle = |1\rangle_b \ |00\rangle_c \ |0\rangle_a = |1000\rangle$$

Quantum Phase Estimation



Wir erhalten:

$$|\Psi_2\rangle = |b\rangle_b |\widetilde{\lambda}\rangle_c |0\rangle_a$$

$$|\Psi_2\rangle = \left(-\frac{1}{\sqrt{2}}|u_0\rangle|01\rangle + \frac{1}{\sqrt{2}}|u_1\rangle|10\rangle\right)|0\rangle_a$$

Ancilla Roation - Eigenwerte invertieren

Wir invertieren das Ancilla Qubit:

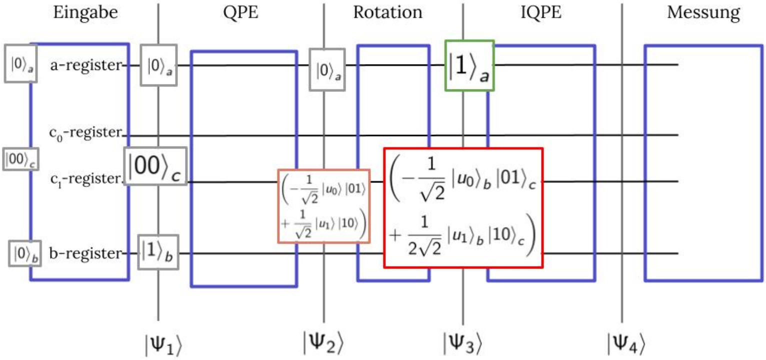
$$|\Psi_3\rangle = \sum_{j=0}^{2^1-1} b_j |u_j\rangle |\widetilde{\lambda}_j\rangle \left(\sqrt{1 - \frac{C^2}{\widetilde{\lambda}_j^2}} |0\rangle + \frac{C}{\widetilde{\lambda}_j} |1\rangle \right)$$

Wir gehen davon aus, dass wir $|1\rangle$ messen.

$$|\Psi_3\rangle = \sqrt{\frac{8}{5}} \left(\frac{1}{\widetilde{\lambda_0}} * -\frac{1}{\sqrt{2}} |u_0\rangle_b |01\rangle_c + \frac{1}{\widetilde{\lambda_1}} * \frac{1}{\sqrt{2}} |u_1\rangle_b |10\rangle_c \right) |1\rangle_a$$

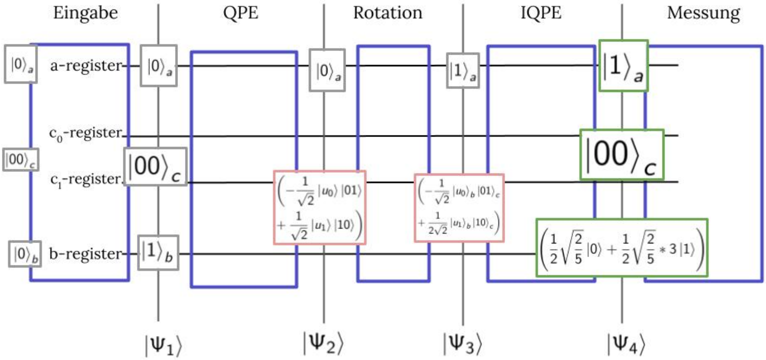
$$|\Psi_3\rangle = \sqrt{\frac{8}{5}} \left(-\frac{1}{\sqrt{2}} |u_0\rangle_b |01\rangle_c + \frac{1}{2\sqrt{2}} |u_1\rangle_b |10\rangle_c \right) |1\rangle_a$$

Ancilla Roation - Eigenwerte invertieren



$$|\Psi_3\rangle = \sqrt{\frac{8}{5}}\left(-\frac{1}{\sqrt{2}}|u_0\rangle|01\rangle|1\rangle + \frac{1}{2\sqrt{2}}|u_1\rangle|10\rangle\right)|1\rangle_a$$

Inverse Quantum Phase Estimation



Wir erhalten:

$$|\Psi_4\rangle = |x\rangle_b \, |00\rangle_c \, |1\rangle_a$$

$$|\Psi_4\rangle = \frac{1}{2}\sqrt{\frac{2}{5}}\left(|0\rangle + 3\,|1\rangle\right)|00\rangle_b \, |1\rangle_a$$

Messung

Um die Wahrscheinlichkeit von x_0 und x_1 zu erhalten, müssen wir ihre Koeffizienten quadrieren

$$Pr[x_0] = \left| \frac{1}{2}\sqrt{\frac{2}{5}} * 1 \right|^2 = \frac{1}{20}$$

$$Pr[x_1] = \left| \frac{1}{2}\sqrt{\frac{2}{5}} * 3 \right|^2 = \frac{9}{20}$$

Das Verhältnis im b-Register ist wie erwartet $1 : 9$.

Gliederung

Laufzeit

Gauß Verfahren

$$\mathcal{O}(N^3)$$

- nicht der schnellste Algorithmus
- gleiche constraints sind zu beachten!!

Laufzeit

Klassisch
Conjugate gradient descent

$$\mathcal{O}(\kappa s log\left(\frac{1}{\epsilon}\right)N)$$

$$\Rightarrow \mathcal{O}(N)$$

Quanten Version
HHL

Laufzeit

Klassisch
Conjugate gradient descent

$$\mathcal{O}(\kappa slog\left(\frac{1}{\epsilon}\right)N)$$

$$\Rightarrow \mathcal{O}(N)$$

Quanten Version
HHL

$$\mathcal{O}(\frac{\kappa^2 s^2}{\epsilon} logN)$$

$$\Rightarrow \mathcal{O}(log(N))$$

Laufzeit

Klassisch
Conjugate gradient descent

$$\mathcal{O}(\kappa s log\left(\frac{1}{\epsilon}\right) N)$$

$$\Rightarrow \mathcal{O}(N)$$

- $N :=$ Anzahl an unbekannten
- $\kappa = \frac{\lambda_{max}}{\lambda_{min}}$: condition number

Quanten Version
HHL

$$\mathcal{O}(\frac{\kappa^2 s^2}{\epsilon} log N)$$

$$\Rightarrow \mathcal{O}(log(N))$$

- $\epsilon :=$ Fehler des Ergebnisses
- $s :=$ s-sparse Matrix: jede Zeile hat max. s Einträge

Laufzeit

Klassisch

Conjugate gradient descent

$$\mathcal{O}(\kappa s log\left(\frac{1}{\epsilon}\right)N)$$

$$\Rightarrow \mathcal{O}(N)$$

Quanten Version

HHL

$$\mathcal{O}(\frac{\kappa^2 s^2}{\epsilon} logN)$$

$$\Rightarrow \mathcal{O}(log(N))$$

Laufzeit

Klassisch

Conjugate gradient descent

$$\mathcal{O}(\kappa s log \left(\frac{1}{\epsilon}\right) N)$$

$$\Rightarrow \mathcal{O}(N)$$

Quanten Version

HHL

$$\mathcal{O}(\frac{\kappa^2 s^2}{\epsilon} log N)$$

$$\Rightarrow \mathcal{O}(log(N))$$

Laufzeit

Klassisch

Conjugate gradient descent

$$\mathcal{O}(\kappa s log\left(\frac{1}{\epsilon}\right) N)$$

$$\Rightarrow \mathcal{O}(N)$$

Quanten Version

HHL

$$\mathcal{O}(\frac{\kappa^2 s^2}{\epsilon} log N)$$

$$\Rightarrow \mathcal{O}(log(N))$$

Takeaway

- exponentialer speed up $\mathcal{O}(N)$ vs $\mathcal{O}(log(N))$
- klassischer algorithmus hat bessere Fehlerabhängigkeit: $log(\frac{1}{\epsilon})$ vs $\frac{1}{\epsilon}$

Einschränkungen

1. niedrige condition number κ

Einschränkungen

1. niedrige condition number κ
2. A muss s-sparse sein

Einschränkungen

1. niedrige condition number κ
2. A muss s-sparse sein
3. nicht jeder Eintrag von $|x\rangle$ auslesbar

Einschränkungen

1. niedrige condition number κ
2. A muss s-sparse sein
3. nicht jeder Eintrag von $|x\rangle$ auslesbar
4. einfache Zustandsvorbereitung des Vektors $\vec{b}$ zum Quantenzustand $|b\rangle$

Einschränkungen

1. niedrige condition number κ
2. A muss s-sparse sein
3. nicht jeder Eintrag von $|x\rangle$ auslesbar
4. einfache Zustandsvorbereitung des Vektors $\vec{b}$ zum Quantenzustand $|b\rangle$
5. Der Ressourcenbedarf ist hoch

Gliederung

Anwendungen

Hauptproblem

- Hauptproblem: gibt keinen vollständigen Vektor aus
- Aber einige Probleme können mit dieser Methode gelöst werden:

Anwendungen

Machine Learning: Least-Square-Fitting

- Datenanpassung mit Least Square Fitting
- durch Berechnung einer Schätzung der inversen Matrix

Anwendungen

Machine Learning: Least-Square-Fitting

- Datenanpassung mit Least Square Fitting
- durch Berechnung einer Schätzung der inversen Matrix

Simulationen von großen Systemen

- Elektrizitätsnetz vielen verbundenen Komponenten
- geringe Anzahl Verbindungen zwischen den Komponenten
- Berechnung des Widerstands durch approximation von Erwartungswerten

Anwendung in IT-Security

HHL in der IT-Security

- in erster Linie nur für Lösen von linearen Systemen
- nicht direkt mit IT-Security verbunden
- aber Potenzial als Subroutine angewendet zu werden

Anwendung in IT-Security

HHL in der IT-Security

- in erster Linie nur für Lösen von linearen Systemen
- nicht direkt mit IT-Security verbunden
- aber Potenzial als Subroutine angewendet zu werden

Mögliche Anwendungen

- secure multi-party computation
- zero-knowledge proofs
- cryptographic key generation and management
- big data analysis/pattern recognition (für Betrugserkennung)

Anwendung in IT-Security

HHL in der IT-Security

- in erster Linie nur für Lösen von linearen Systemen
- nicht direkt mit IT-Security verbunden
- aber Potenzial als Subroutine angewendet zu werden

Mögliche Anwendungen

- secure multi-party computation
- zero-knowledge proofs
- cryptographic key generation and management
- big data analysis/pattern recognition (für Betrugserkennung)

Es wäre wichtig, mehr Anwendungen zu finden, welche den Anforderungen entsprechen.

Optimierungen

Modifikationen und Optimierung

- QRAM zur Vorbereitung von $|b\rangle$

Optimierungen

Modifikationen und Optimierung

- QRAM zur Vorbereitung von $|b\rangle$
- kein Ancilla-Bit erforderlich unter bestimmten Voraussetzungen

Optimierungen

Modifikationen und Optimierung

- QRAM zur Vorbereitung von $|b\rangle$
- kein Ancilla-Bit erforderlich unter bestimmten Voraussetzungen
- Variable time amplitude amplification um condition number κ zu verbessern

Perspektive

- Großer Einfluss im Bereich Quantum Machine Learning

Perspektive

- Großer Einfluss im Bereich Quantum Machine Learning
- noch keine bahnbrechenden Anwendungen (wie z.B. Shors Algorithmus zum Brechen von RSA)

Perspektive

- Großer Einfluss im Bereich Quantum Machine Learning
- noch keine bahnbrechenden Anwendungen (wie z.B. Shors Algorithmus zum Brechen von RSA)
- aber viel aktive Forschung um neue Verbesserungen im Algorithmus zu finden

Perspektive

- Großer Einfluss im Bereich Quantum Machine Learning
- noch keine bahnbrechenden Anwendungen (wie z.B. Shors Algorithmus zum Brechen von RSA)
- aber viel aktive Forschung um neue Verbesserungen im Algorithmus zu finden
- zeigt deutlichen Fortschritt in der Quantencomputing Welt