Quantum Implementation and Optimization of the HHL Algorithm

Dhruv Sood
Collaborators : Nilmani Mathur and Vikram Tripathi

June 24, 2026

Introduction

- This project focuses on methods of solving the Quantum Linear Systems Problem (QLSP), primarily via the Harrow-Hassidim-Lloyd (HHL) algorithm[1]. When implemented on quantum hardware, it is able to solve a linear system of equations in $O\left(\frac{\kappa^2 s^2}{\epsilon} \ln N\right)$ steps, which is an exponential speed-up compared to classical methods where the scaling is at best $\sim N^{2.81}$).

[1] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd, Phys. Rev. Lett. 103 (2009)

Introduction

- This project focuses on methods of solving the Quantum Linear Systems Problem (QLSP), primarily via the Harrow-Hassidim-Lloyd (HHL) algorithm[1]. When implemented on quantum hardware, it is able to solve a linear system of equations in $O\left(\frac{\kappa^2 s^2}{\epsilon} \ln N\right)$ steps, which is an exponential speed-up compared to classical methods where the scaling is at best $\sim N^{2.81}$).
- Linear Systems of equations are ubiquitous in all fields of science. A more efficient procedure of solving them has the potential to drastically accelerate studies of physical systems.

[1] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd, Phys. Rev. Lett. 103 (2009)

Introduction

- This project focuses on methods of solving the Quantum Linear Systems Problem (QLSP), primarily via the Harrow-Hassidim-Lloyd (HHL) algorithm[1]. When implemented on quantum hardware, it is able to solve a linear system of equations in $O\left(\frac{\kappa^2 s^2}{\epsilon} \ln N\right)$ steps, which is an exponential speed-up compared to classical methods where the scaling is at best $\sim N^{2.81}$).
- Linear Systems of equations are ubiquitous in all fields of science. A more efficient procedure of solving them has the potential to drastically accelerate studies of physical systems.
- The relevant literature, though extensive, does not touch upon its implementation on a quantum computer/simulator for arbitrary linear systems $\mathbf{A}\vec{x} = \vec{b}$ in a scalable manner. This is the focus of this work.

[1] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd, Phys. Rev. Lett. 103 (2009)

The HHL Algorithm

- HHL consists of subroutines :
  - State Preparation
  - Quantum Phase Estimation
  - Ancillary Rotations
  - Inverse Quantum Phase Estimation
  - Measurement and Post-Selection

The HHL Algorithm

- HHL consists of subroutines :
  - State Preparation
  - Quantum Phase Estimation
  - Ancillary Rotations
  - Inverse Quantum Phase Estimation
  - Measurement and Post-Selection
- QPE is the most expensive of these, requiring multiple applications of a controlled e^{iAt}.

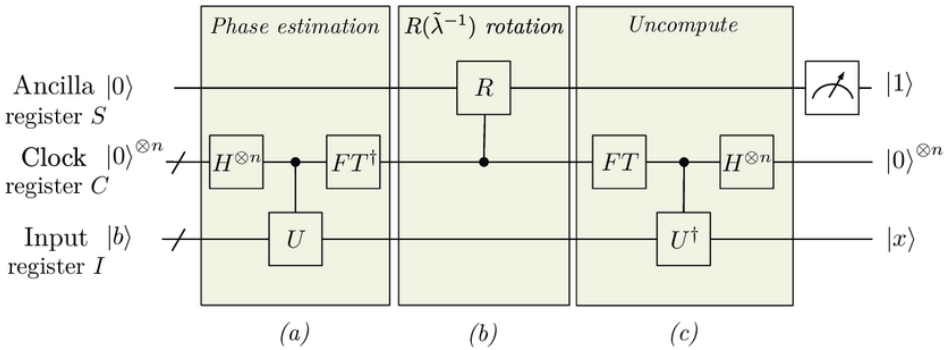


Figure: Block Diagram of the HHL Algorithm

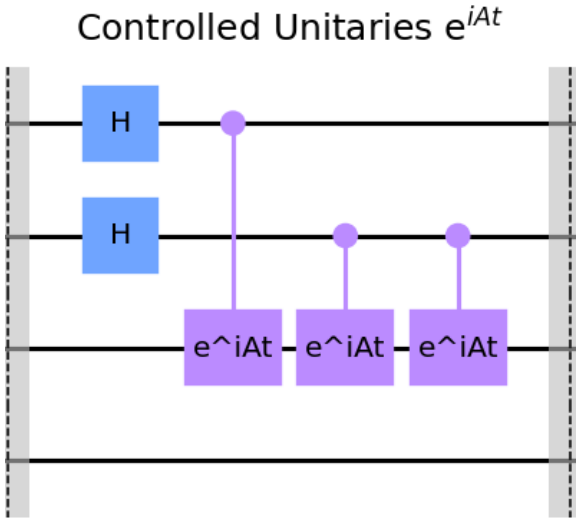


Figure: QPE for a N=2 System

Adiabatic Evolution

- The other method we have tried leverages adiabatic evolution to evolve the state of the qubits qubits.

[2] Pedro C.S. Costa, Dong An, Yuval R. Sanders, Yuan Su, Ryan Babbush, and Dominic W. Berry, PRX Quantum 3 (2022), 040303.

Adiabatic Evolution

- The other method we have tried leverages adiabatic evolution to evolve the state of the qubits qubits.
- We set up the evolution such that the input state $|b\rangle$ is the ground state of some initial Hamiltonian H_0, while the desired state $|x\rangle$ is that of the final Hamiltonian H_1.

[2] Pedro C.S. Costa, Dong An, Yuval R. Sanders, Yuan Su, Ryan Babbush, and Dominic W. Berry, PRX Quantum 3 (2022), 040303.

Adiabatic Evolution

- The other method we have tried leverages adiabatic evolution to evolve the state of the qubits qubits.
- We set up the evolution such that the input state $|b\rangle$ is the ground state of some initial Hamiltonian H_0, while the desired state $|x\rangle$ is that of the final Hamiltonian H_1.
- Explicitly these are,[2]

$$H(s) = (1-s)H_0 + sH_1$$

, with

$$H_0 := \begin{pmatrix} 0 & Q_b \\ Q_b & 0 \end{pmatrix} \text{ and } H_1 := \begin{pmatrix} 0 & AQ_b \\ Q_bA & 0 \end{pmatrix}$$

and $Q_b = \mathbf{1} - |b\rangle\langle b|$.

[2] Pedro C.S. Costa, Dong An, Yuval R. Sanders, Yuan Su, Ryan Babbush, and Dominic W. Berry, PRX Quantum 3 (2022), 040303.

Methods: Trotterisation

- The first method we tried was writing A as sum of simpler matrices using the standard trotterisation procedure.[3]

$$e^{h\sum_{k=1}^{\Lambda} A_k + \mathcal{O}\left(h^{n+1}\right)} = \left(\prod_{k=1}^{\Lambda} e^{A_k c_1 h}\right)\left(\prod_{k=1}^{\Lambda} e^{A_k d_1 h}\right)\cdots\left(\prod_{k=1}^{\Lambda} e^{A_k c_q h}\right)\left(\prod_{k=\Lambda}^{1} e^{A_k d_q h}\right) \quad (1)$$

with the coefficients being computable using the commutators of the A_k.

[3] A. A. Avtandilyan and W. V. Pogosov, Quantum Information Processing 24 (2024), no. 1, 8.

Methods: Trotterisation

- The first method we tried was writing A as sum of simpler matrices using the standard trotterisation procedure.[3]

$$e^{h\sum_{k=1}^{\Lambda} A_k + \mathcal{O}\left(h^{n+1}\right)} = \left(\prod_{k=1}^{\Lambda} e^{A_k c_1 h}\right)\left(\prod_{k=1}^{\Lambda} e^{A_k d_1 h}\right)\cdots\left(\prod_{k=1}^{\Lambda} e^{A_k c_q h}\right)\left(\prod_{k=\Lambda}^{1} e^{A_k d_q h}\right) \quad (1)$$

  with the coefficients being computable using the commutators of the A_k.

- With trotterisation it is generally known that while the number of trotter steps introduced decreases the error introduced, it also increases the circuit depth drastically and increases runtime.

[3] A. A. Avtandilyan and W. V. Pogosov, Quantum Information Processing 24 (2024), no. 1, 8.

Methods: Trotterisation

- The first method we tried was writing A as sum of simpler matrices using the standard trotterisation procedure.[3]

$$e^{h\sum_{k=1}^{\Lambda} A_k + \mathcal{O}\left(h^{n+1}\right)} = \left(\prod_{k=1}^{\Lambda} e^{A_k c_1 h}\right)\left(\prod_{k=1}^{\Lambda} e^{A_k d_1 h}\right)\cdots\left(\prod_{k=1}^{\Lambda} e^{A_k c_q h}\right)\left(\prod_{k=\Lambda}^{1} e^{A_k d_q h}\right) \quad (1)$$

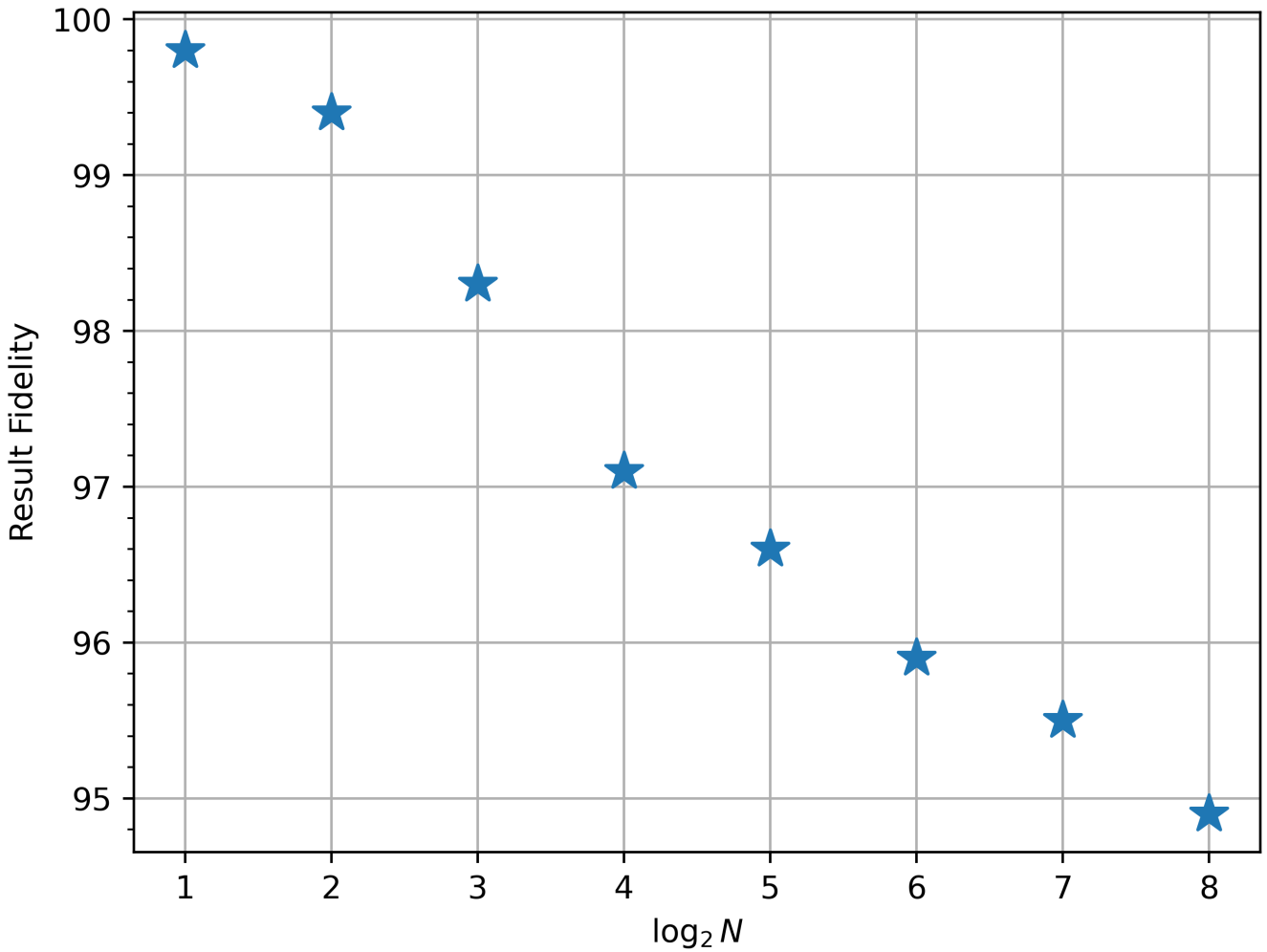
  with the coefficients being computable using the commutators of the A_k.

- With trotterisation it is generally known that while the number of trotter steps introduced decreases the error introduced, it also increases the circuit depth drastically and increases runtime.

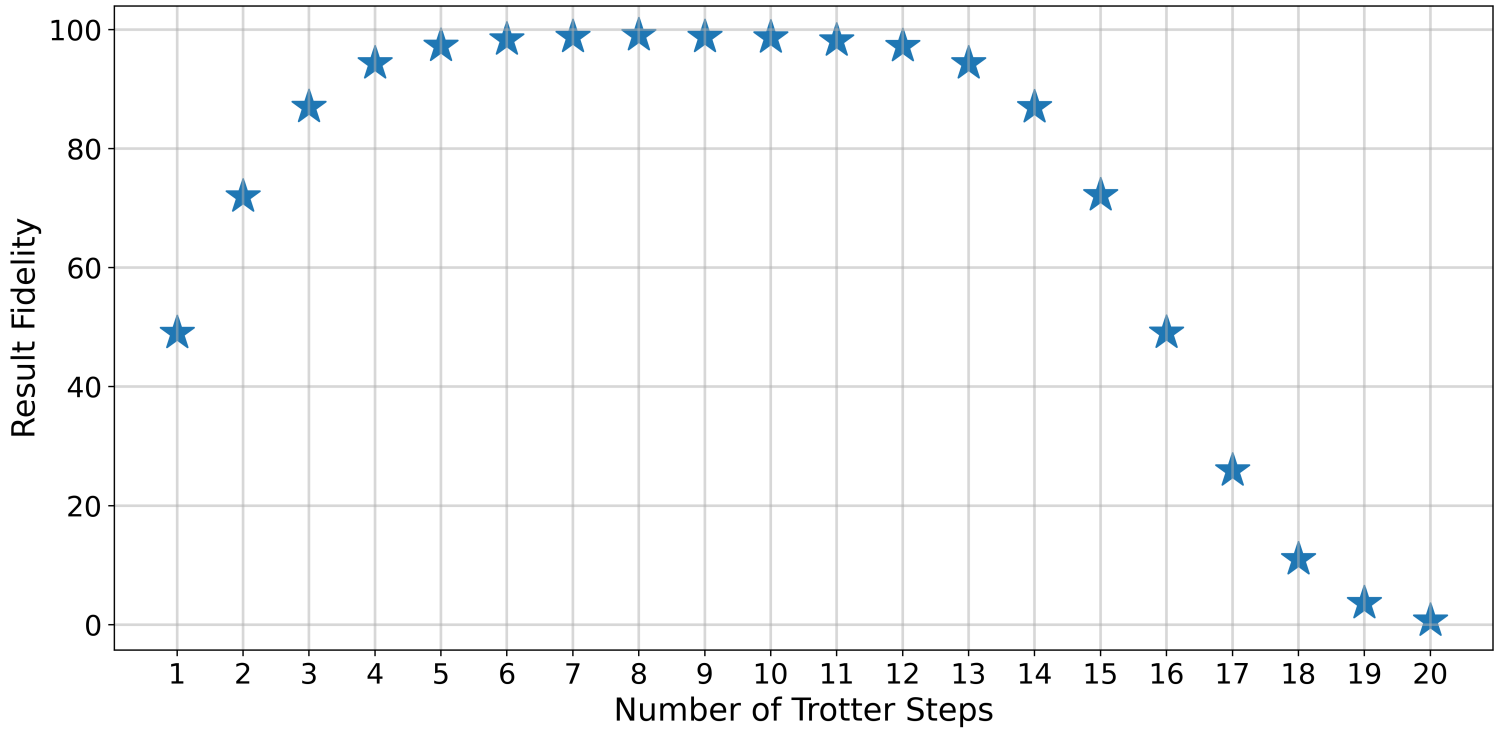
- For larger systems trotterisation decreased the fidelity and we found that for mid-sparsity trotterisation seems to the be optimal way.

[3] A. A. Avtandilyan and W. V. Pogosov, Quantum Information Processing 24 (2024), no. 1, 8.

Trotterisation Fidelity



Number of Trotter Steps



Methods: Block Encoding

- Block encoding involves embedding the matrix A into a higher dimensional matrix constructed such that the resulting operator is unitary.

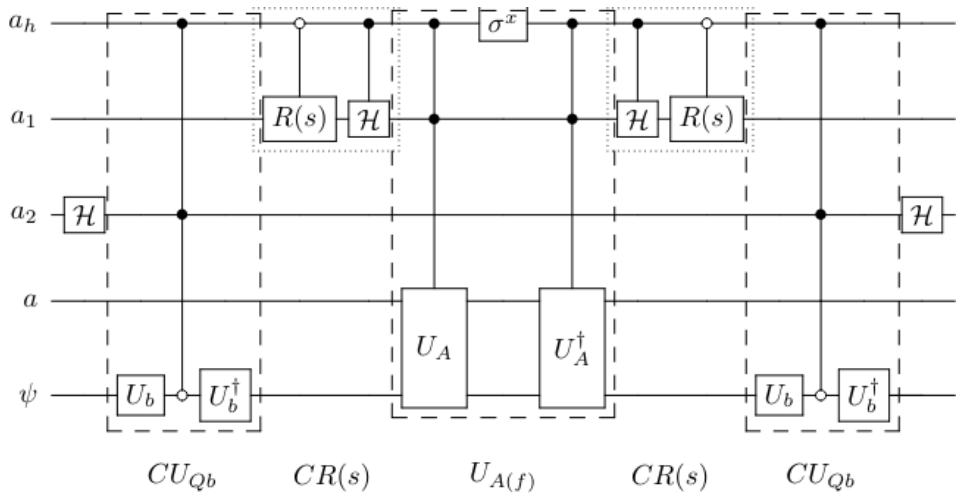


Figure: Block Encoding of H(s)

Methods: Block Encoding

- Block encoding involves embedding the matrix A into a higher dimensional matrix constructed such that the resulting operator is unitary.
- This was used primarily for constructing the evolution operator for Adiabatic method by using a set of ancillary qubits to rotate between H_0 and H_1 to simulate the application of $H(s)$ on the register.

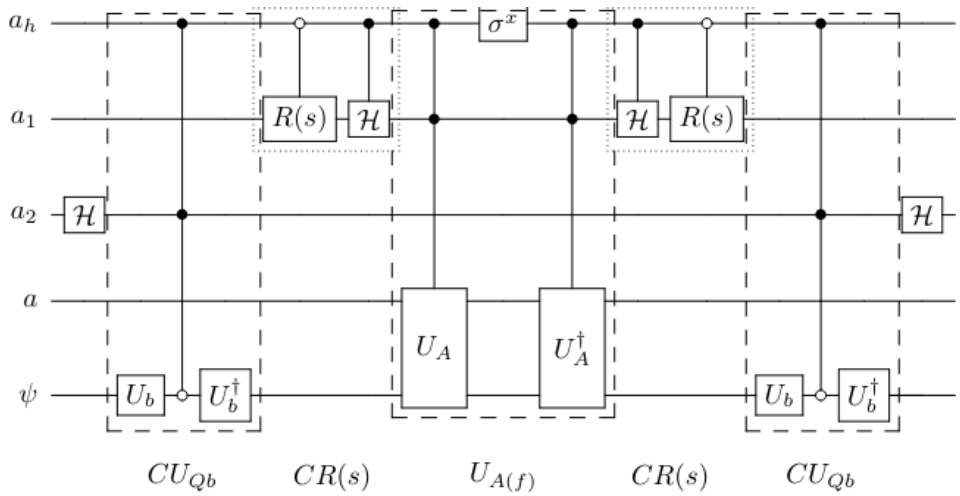


Figure: Block Encoding of H(s)

Methods: Block Encoding

- Block encoding involves embedding the matrix A into a higher dimensional matrix constructed such that the resulting operator is unitary.
- This was used primarily for constructing the evolution operator for Adiabatic method by using a set of ancillary qubits to rotate between H_0 and H_1 to simulate the application of $H(s)$ on the register.
- Using this method increases the number of qubits required and limits the system size but gives higher fidelity results than trotterisation.

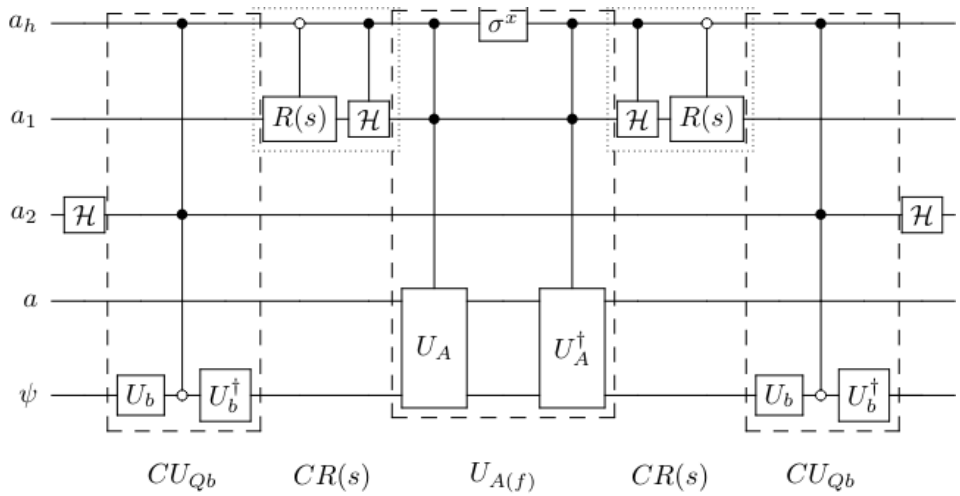
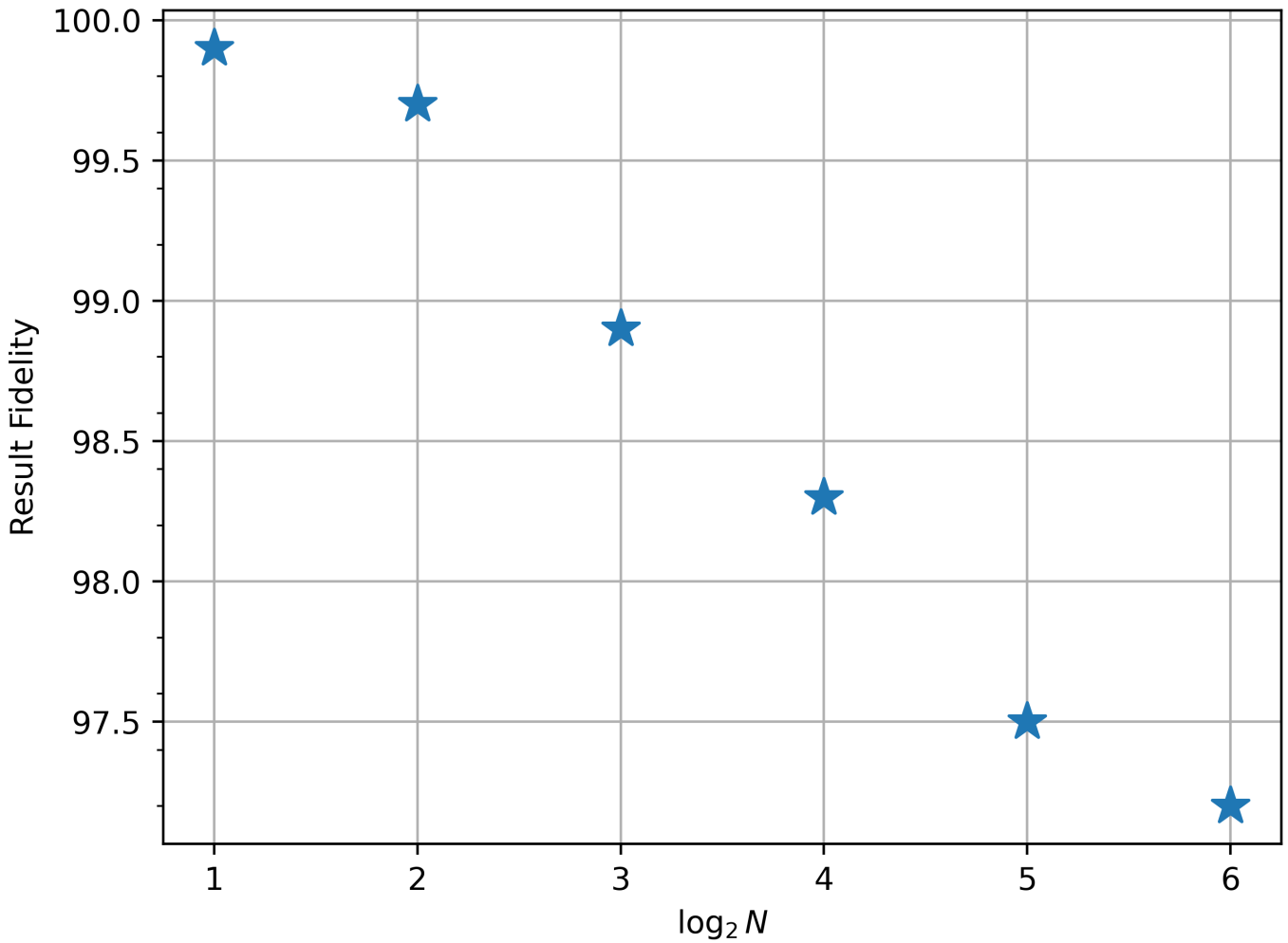


Figure: Block Encoding of H(s)

Block Encoding Fidelity



Applications to Differential Equations

Finite-difference discretisation converts PDEs into sparse linear systems.

Studied examples:

1. Heat Equation
2. Wave Equation
3. Poisson Equation

Observed fidelities:

Equation	System Size	Fidelity
Heat Equation	$N = 1024$	90.8%
Wave Equation	$N = 1024$	88.1%
Poisson Equation	$N = 256$	93.4%

Result fidelity for the Heat Equation

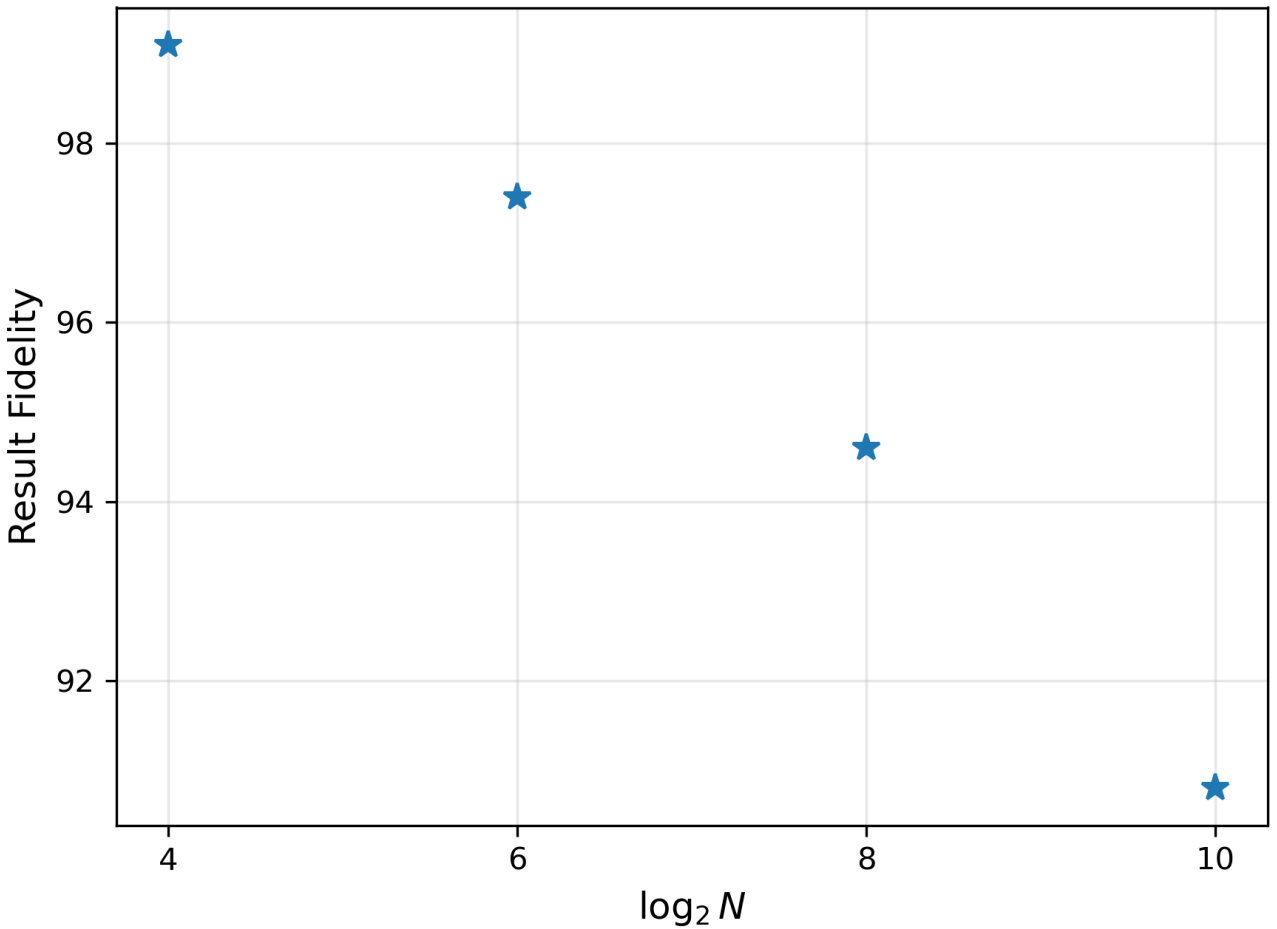


Figure: Result fidelity for the Heat Equation as a function of lattice size.

Result fidelity for the Wave Equation

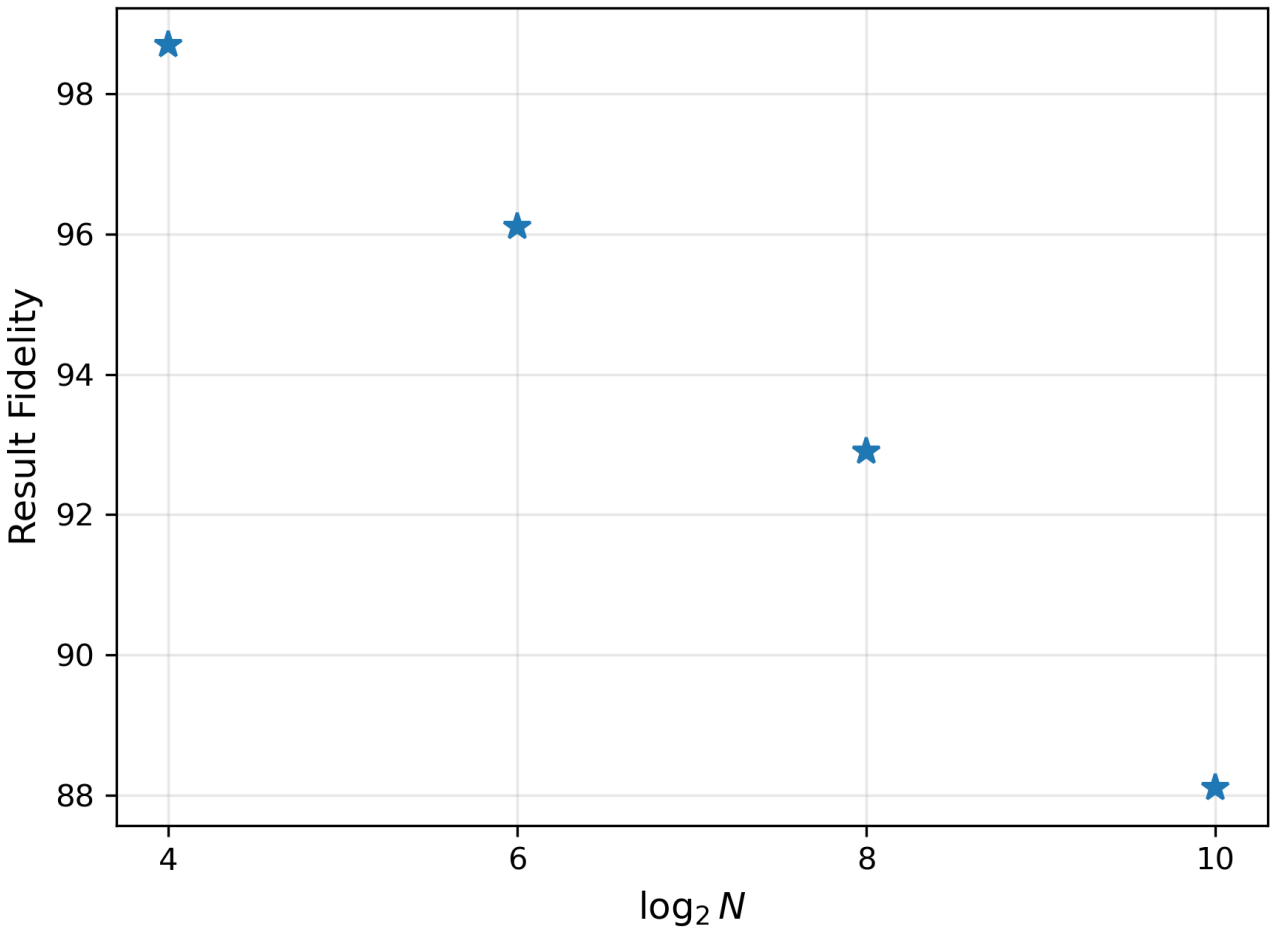


Figure: Result fidelity for the Wave Equation as a function of lattice size.

Result fidelity for the Poisson Equation

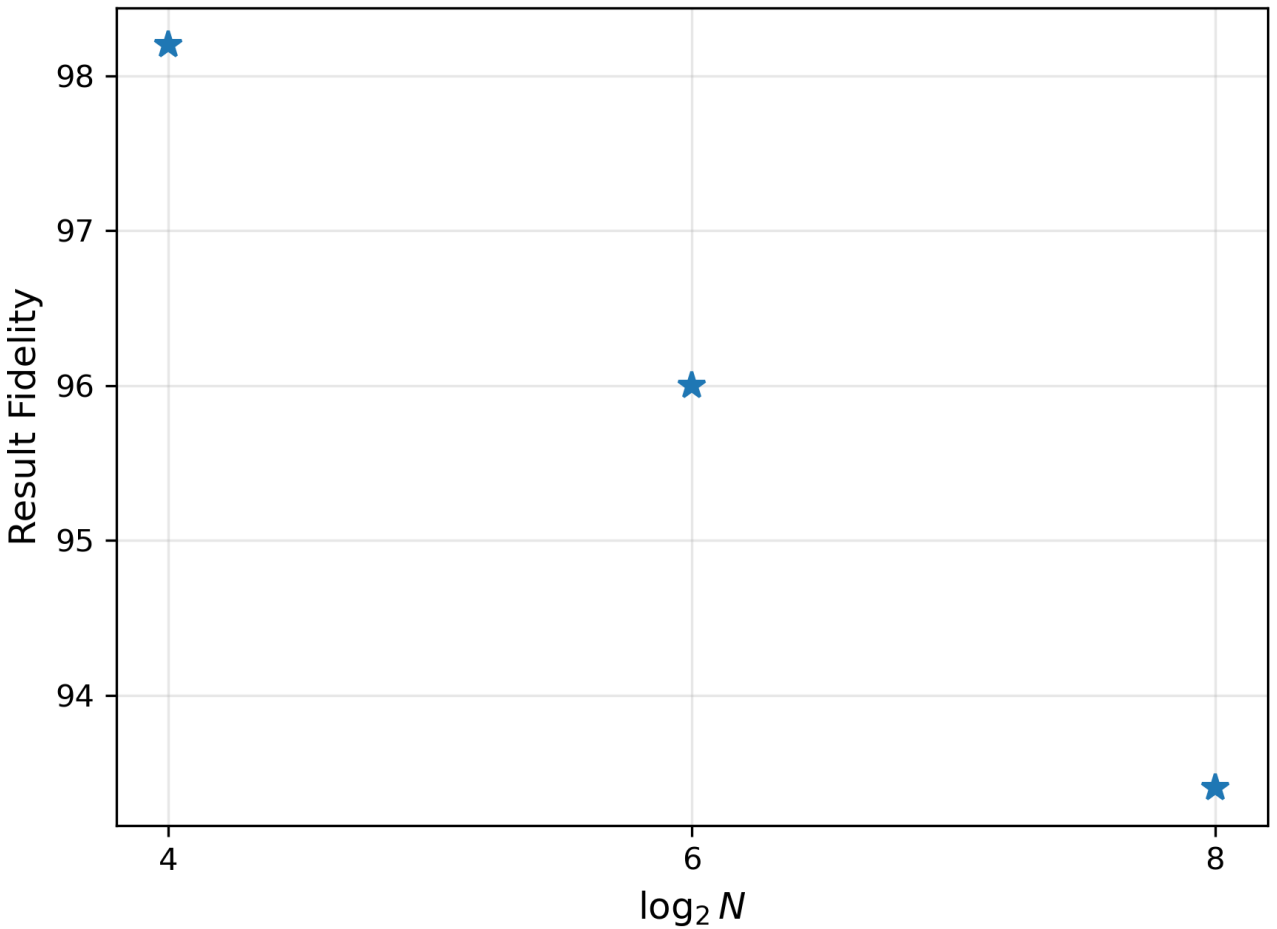


Figure: Result fidelity for the Poisson Equation as a function of lattice size.

Condition Number Dependence

Condition number:

$$\kappa(A) = \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}}$$

Relevance :

- Eigenvalue inversion amplifies small eigenvalue errors.
- Ancilla post-selection probability decreases approximately as

$$P_{\text{success}} \propto \kappa^{-2}.$$

Observed trend:

- Fidelities remain high for moderate κ,
- but drop rapidly beyond roughly

$$\kappa \sim 2^{10}.$$

Result fidelity as a function of the Condition Number

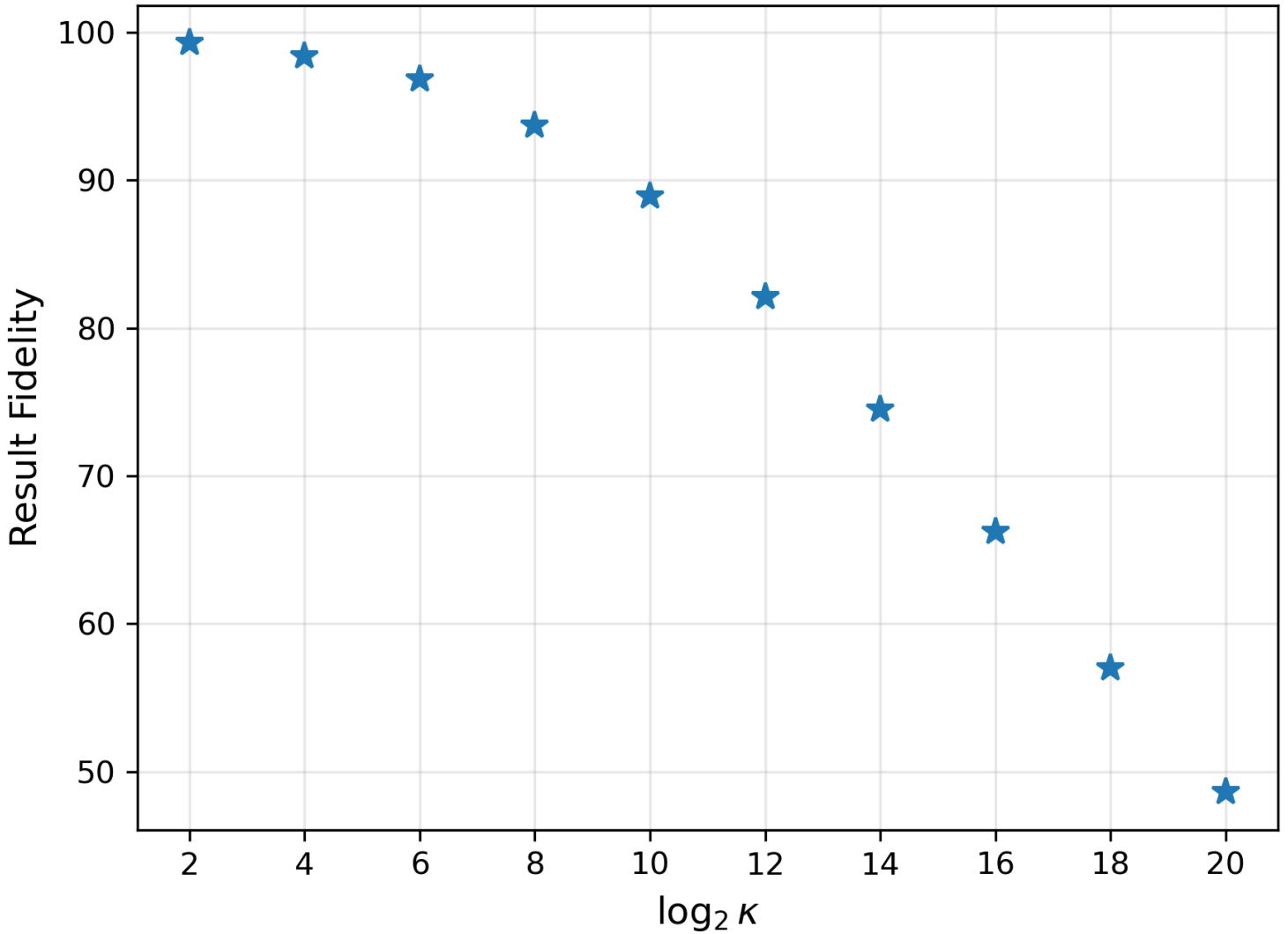


Figure: Result fidelity as a function of the condition number for $N = 256$

Results

- For a diagonal matrix A, we are only limited by the memory/number of qubits and time available. Systems upto $N = 1024$ have been run and we have obtained an average fidelity of 99.3%.

Results

- For a diagonal matrix A, we are only limited by the memory/number of qubits and time available. Systems upto $N = 1024$ have been run and we have obtained an average fidelity of 99.3%.
- For tridiagonal matrices, highest N achieved is 256 with a fidelity of 94.9%.

Results

- For a diagonal matrix A, we are only limited by the memory/number of qubits and time available. Systems upto $N = 1024$ have been run and we have obtained an average fidelity of 99.3%.
- For tridiagonal matrices, highest N achieved is 256 with a fidelity of 94.9%.
- For $< N/2$-dense matrices I have been able to go upto $N = 32$ with a fidelity of 91.7%.

Results

- For a diagonal matrix A, we are only limited by the memory/number of qubits and time available. Systems upto $N = 1024$ have been run and we have obtained an average fidelity of 99.3%.
- For tridiagonal matrices, highest N achieved is 256 with a fidelity of 94.9%.
- For $< N/2$-dense matrices I have been able to go upto $N = 32$ with a fidelity of 91.7%.
- Finally for dense matrices, the system sizes $N = 16$ and $N = 32$ can be respectively solved with average fidelities 90.5% and 80.3%.