Variational Quantum Algorithm for the Poisson eqn

February 17, 2025 5:11 PM

Introduction

→ The Poisson Eqn appears in various scientific fields, such as quantum mechanics & Markov chains

→ Existing quantum algorithms for solving the Poisson eqn require fault-tolerant quantum computers, which are not yet practical

→ The paper introduces a variational quantum algorithm (VQA) that can run on noisy intermediate-scale quantum (NISQ) devices

The Variational Quantum Algorithm (VQA) ✶ - Very Important - can be implemented on the computers of today

→ The Poisson eqn is first discretized using the finite-difference method, converting it into a linear system (recalling that quantum computers can only run linear systems)

→ The coefficient matrix of this linear system is then decomposed into a tensor product of simple operators

→ The decomposition reduces the # of required quantum measurements, making the approach more efficient.

Discretization of the Poisson Eqn

→ The d-dimension Poisson eqn w/ Dirichlet boundary conditions is given by:

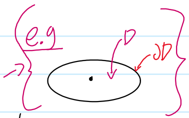
$$-\Delta u(x) = S(x), \quad x \in D$$

w/ boundary conditions

$$u(x) = 0, \quad x \in \partial D$$

↳ boundary of the Domain

So x is of the boundary values of the domain, D.



Where:

Δ = the Laplace operator

$D = (0,1)^d$ is the domain

$S(x)$ = a given fct^n

After discretization, the Poisson eqn becomes the linear system:

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

where the coefficient matrix A is the n-dimensional tri-diagonal matrix:

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & -1 & 2 & -1 \\ 0 & 0 & \cdots & 0 & -1 & 2 \end{bmatrix}$$

In the one-dimension (1D) case

for the discretized Poisson eqn. It represents how the system behaves when solving a one-dimensional Poisson eqn numerically.

Now, when we extend this to higher dimensions (d-dimensions), we must account for interactions in multiple directions. The coefficient matrix $A^{(d)}$ must include information about interactions along all dimensions.

Now, when we extend this to higher dimensions (d-dimensions), we must account for interactions in multiple directions. The coefficient matrix $A^{(d)}$ must include information about interactions along all dimensions.

For a d-dimensional matrix, the coefficient matrix is:

$$\boxed{A^{(d)} = A \otimes I \otimes \cdots \otimes I + I \otimes A \otimes I \otimes \cdots \otimes I + \cdots + I \otimes \cdots \otimes I \otimes A}$$

where:

$\otimes$ is the Kronecker Product → This represents a tensor (block) product b/twn matrices. It is used to extend a 1-D system matrix into higher dimensions

① $A \otimes I \otimes \cdots \otimes I$ → Describes interactions along the first dimension

② $I \otimes A \otimes I \otimes \cdots \otimes I$ → Describes interactions along the second dimension

③ $I \otimes \cdots \otimes I \otimes A$ → Describes interactions along the d-th dimension

Each term ensures that the discrete Laplacian operator acts independently in each dimension

From a Quantum Computing Perspective

In quantum computing, the Poisson eqn is often mapped to quantum states. The goal is to solve:

$$\boxed{A^{(d)} \vec{x} = \vec{b}}$$

where x is encoded as a quantum state. By efficiently decomposing $A^{(d)} x = b$ into tensor products, we reduce the complexity of the problem, making it suitable for quantum variational algorithms (VQAs).

Why is This Important?

→ This structure allows us to efficiently represent multi-dimensional Poisson eqns as a linear system

→ Discretizing the Poisson eqns as linear system helps in solving phyisics & engineering problems that require solving differential eqns in multiple dimensions (e.g. heat diffusion, fluid dynamics, & electromagnetics).

→ Lastly the Kronecker product formulation simplifies computations & allows for efficient implementation on quantum computers.

Variational Quantum Algorithm (VQA) for Poisson Eqn

→ The solⁿ $|x\rangle$ (ket vector, x) is obtained by minimizing the energy fctⁿ of the Hamiltonian:

$$H = A^{*}(I - |b\rangle\langle b|)A$$

Also can be written as $A^{\dagger}$ or A^{H}; the Hermitian adjoint of A

(a) A^{*} is the Hermitian conjugate (the complex conjugate transpose) of A

Now, recall, the adjoint (or adjugate) is specifically the transpose of the cofactor matrix of a given matrix

↳ So it's not just any transpose, it's the conjugate/transpose of the cofactor matrix of a given linear system

Not to be confused w:

→ The Symmetric Matrix: where $A^{T} = A$, &;

→ The Hermitian Matrix: where A^{*} (or A^{H}) $= A$

The Cofactor Matrix

The cofactor matrix is a matrix where each element is the cofactor of the corresponding element in the original matrix.

-> It plays an important role in finding the inverse of a matrix using the adjugate (adjoint) method

Definition: For an nxn matrix, the cofactor matrix, C, is formed by taking the cofactor of each element A.

The Cofactor: is found by taking the minor of each element & multiplying it by the cofactor multiplier $(-1)^{i+j}$, the sign factor, which the sign alternates based on the row & column indices.

minor: the determinant of the submatrix obtained by removing the ith row & jth column containing the element

$\bar{A}, A^*$ - Complex Conjugate
$A^H, A^\dagger$ - Complex Conjugate Transpose

-> Note, if A is a real symmetric matrix; $A^\dagger = A$

↳ Adjoint: (operator theory) -> Complex Conjugate Transpose

ket bra, bra ket $\boxed{\langle b| \; |b\rangle} = \langle b|b\rangle = \|b\|_2^2$

(b) $|b\rangle\langle b|$ The Projection Operator

-> $|b\rangle$ is a quantum state representing the right-hand side of the eqn $A\vec{x} = \vec{b}$

ket b, bra b => matrix operator

-> The term $|b\rangle\langle b|$ is a projection operator, which projects any vector onto $|b\rangle$

outer product

-> This ensures that our sol^n is constrained to the subspace of $|b\rangle$

$|b\rangle\langle b|$ - projection operator

-> $|b\rangle\langle b|x\rangle$

$|b\rangle\underbrace{\langle b|x\rangle}_{\text{scalar; inner product btwn } b \,\&\, x}$

The outer product (ex. $\underbrace{|a\rangle\langle b|}_{\text{operator}}$) produces a matrix.

where $a, b \in \mathbb{R}$

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \cdot \begin{pmatrix} b_1 & b_2 & b_3 \end{pmatrix}$$

$$\underbrace{\begin{pmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \end{pmatrix}}_{3\times 3 \text{ matrix}}$$

-> we have now projected $|x\rangle$ onto the direction $|b\rangle$

-> operations can be performed on nx1 · 1xm matrix where n≠m & m=n

Inner Product -> Combine 2 vectors s.t. you get a scalar.
Outer Product -> Combine 2 vectors s.t. you get a matrix (operator).

(c) $I - |b\rangle\langle b|$ (Projecting Away from $|b\rangle$)

-> The identity matrix, I, represents all possible directions in the vector space.

-> The term $(I - |b\rangle\langle b|)$ removes the component of any vector aligned w $|b\rangle$.

-> This means we're looking for a solution that is perpendicular to $|b\rangle$ in some sense

orthogonal complement of $|b\rangle$

(d) Final Form: $H = A^\dagger(I - |b\rangle\langle b|)A$

↘ we want to minimize this; as the sol^n is across $|b\rangle$
↳ costs energy; loss of energy

-> This eqn defines a Hamiltonian

What is a Hamiltonian? (H in Quantum Mechanics & Linear Systems) -> observiable (Hermitian) Operator

-> The Hamiltonian (H) is a fundamental operator in quantum mechanics that represents the total energy of the system. It plays a crucial role in evolution of quantum states & in solving quantum algorithms like the Variational Quantum Algorithm (VQA).

-> In the context of the Variational Quantum Algorithm (VQA) for solving linear systems, the Hamiltonian is constructed to encode the solution of the eqn:

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

where:
A = the coefficient matrix (from discretizing the Poisson eqn).
x = the sol^n vector
b = the right-hand side (RHS) vector

The Def. as Described in the Paper

The paper defines the Hamiltonian as:

$$H = A^\dagger(I - |b\rangle\langle b|)A$$

This formulation ensures that the lowest energy (ground state) of H corresponds to the sol^n of the linear system $Ax = b$.

Reviewing the terms:

Reviewing the terms:

$A^{\dagger}$: The Hermitian (complex-conjugate) transpose of A.

$I - |b\rangle\langle b|$: A projection operator that removes the component of A along $|b\rangle$

Where the result ensures that the ground state of H corresponds to the correct solⁿ x.
↳ lowest energy possible
$x = H_G$ (H ground state)

2. The General meaning of Hamiltonian in Quantum Mechanics

In quantum mechanics, the Hamiltonian is an operator that governs the time evolution of a quantum state $|\psi\rangle$ according to Schrödinger's eqn:

$$\boxed{i\hbar \frac{d}{dt}|\psi(t)\rangle = H|\psi(t)\rangle}$$

↳ i is the imaginary unit (also denoted j)

$\hat{i}, \hat{j}, \hat{k}$

z ($\hat{k}$), ($\hat{j}$) y, x ($\hat{i}$)

Where:

-> H represents the total energy (kinetic + potential) of the system

-> $|\psi(t)\rangle$ is the quantum state vector @ a time t

-> $\hbar$ is Plank's constant divided by 2π

In quantum computing, we construct Hamiltonians to encode problems, such that the ground state of H represents the optimal solⁿ.

3. Why is the Hamiltonian Important in VQA?

-> The Variational Quantum Algorithm (VQA) minimizes H to find a solⁿ x

-> Instead of solving $A\vec{x} = \vec{b}$ directly, we find a quantum state $|\psi(\theta)\rangle$ that minimizes the expectation value:
↳ energy value (cost)

$$\boxed{E(\theta) = \langle\psi(\theta)|H|\psi(\theta)\rangle}$$

-> By tuning the quantum circuit parameters θ, the algorithm finds the lowest energy state of H, which corresponds to the solⁿ of $Ax = b$.

4. Key Takeaways

☑ The Hamiltonian (H) represents the total energy in quantum mechanics

☑ In VQA, H is constructed such that its ground state corresponds to the solⁿ x of the linear system

☑ The goal of VQA is to minimize H using a quantum-classical hybrid approach, tuning parameters θ to approximate the solⁿ.

~

The trial quantum state is:

$$|\psi(\theta)\rangle = U(\theta)|0\rangle^{\otimes n}$$

-> thus $\otimes n$ -> tensor product -> involves anything

-> $|0\rangle$ tensored n times

↳ $|0\rangle$ represents 1 qubit

Almost the same thing, but

operator A	: matrix A	
flexible coordinate sys (independent of coord sys)	more specific coordinate sys	already choose the coordinate system ↳ depends on the coordinate system

ex. Cartisan A & polar A => Same operator A, but different matrices A

Cartisian A & spherical A
↳ same operators of A but different matrices of A

Kronker product -> involves the tensor of

A_{op} B_{op}

tensor

$A_{op} \otimes B_{op}$

↳ abstract op

A_m B_m -> know A_{op} & B_{op}

Q: what does the matrix rep of A_{op} & B_{op} look like?

$A_m \otimes B_m$

↳ Kronker product -> corresponds to the matrix rep of $A_{op} \otimes B_{op}$ ↳ tensor

$|\psi(\theta)\rangle = U(\theta)|0\rangle$

↳ $|0\rangle$ represents 1 qubit

Cartisian A & spherical A
↳ same operators of A but different matrices of A

different matrices A

Kronker product → involves the tensor of

$A_{op} \otimes B_{op}$
↳ tensor

*where $U(\theta)$ is the parameterized quantum circuit.

The Trial Quantum State $|\psi(\theta)\rangle = U(\theta)|0\rangle^{\otimes n}$ represents how a quantum state is prepared using a parametized quantum circuit.

Q: What Does This Eqn Mean? $\boxed{|\psi(\theta)\rangle = U(\theta)|0\rangle}$

- $|\psi(\theta)\rangle$: This is the trial quantum state, which is an approximation of the desired sol^n
- $|0\rangle$: This is the initial state of the quantum system, which is typically set to all zero qubits in quantum computation
- $U(\theta)$: This is a parameterized quantum circuit, meaning it contains adjustible parameters θ that can be optimized

The goal of the variational algorithm is to adjust θ s.t. $|\psi(\theta)\rangle$ is as close as possible to the actual sol^n to the given problem.

2. What is a Parametized Quantum Circuit $U(\theta)$?

A parametized quantum circuit is a quantum gate sequence that depends on a set of classical parameters θ.

Mathematically: $U(\theta) = U_L(\theta_L)\ldots U_2(\theta_2)U_1(\theta_1)$

where:

→ Each $U_i(\theta_i)$ is a quantum gate that depends on some parameter θ_i.

→ The circuit is trained by adjusting θ iteratively

☞ This is similar to how neural networks in machine learning adjust their weights to minimize error!

Example of a Simple Parameterized Quantum Circuit

→ A common choice for $U(\theta)$ is the Quantum Approximate Optimization Ansatz (QAOA) or Hardware-Efficient Ansatz:

$$\boxed{U(\theta) = e^{-iH_{problem}\theta_p} e^{-iH_{mixer}\theta_m}}$$

where:

→ $H_{problem}$: encodes the problem constraints

→ H_{mixer}: ensures exploration of the sol^n space

Q: Why do we use this approach?

1. Quantum State Preparation

→ The initial state $|0\rangle^{\otimes n}$ is easy to prepare

→ The quantum circuit $U(\theta)$ transforms it into a more complex state that hopefully approximates the sol^n

2. Optimization Process

approximate sol^n

2. Optimization Process

-> We optimize θ to minimize an energy fct^n (called the cost fct^n):

$$\boxed{E(\theta)=\langle\psi(\theta)|H|\psi(\theta)\rangle}$$ The cost fct^n

-> This fct^n is evaluated using a quantum computer, while the optimization of θ is done on a classical computer

3. Flexibility in Quantum Algorithms

-> Different quantum circuits $U(\theta)$ can be designed for different problems
-> Variational algorithms allow quantum computers to solve problems on noisy hardware (NISQ devices)

4. Summary

- [x] $|\psi(\theta)\rangle$ is a trial quantum state, which is generated by applying a parameterized quantum circuit $U(\theta)$ to an initial state $|0\rangle$
- [x] The parameters θ are adjusted iteratively using classical optimization to find the best sol^n
- [x] This approach allows hybrid quantum-classical computation, making it useful for solving linear systems, optimization problems, & quantum chemistry

The cost fct^n to minimize is $E(\theta)=\langle\psi(\theta)|H|\psi(\theta)\rangle$

-> where a classical optimizer (such as gradient descent) is used to adjust parameters θ to minimize $E(\theta)$.
-> The quantum sol^n, $|x\rangle$ is given by:

$$|x\rangle \approx |\psi(\theta_{opt})\rangle$$

where θ_{opt} is the optimized parameter.

Decomposition of the Coefficient Matrix A

-> We are introduced to the tensor product decomposition of A, which reduces quantum resource requirements.

-> The decomposition is done using simple operators: $\{I, \sigma_+, \sigma_-\}$

where: $\sigma_+ = |0\rangle\langle 1|$ & $\sigma_- = |1\rangle\langle 0|$

-> The final decomposition has only $2\log_2 n + 1$ terms, compared to exponential growth in standard methods.

Understanding the Tensor Decomposition of A

$$\boxed{A = I \otimes A_{m-1} - \sigma_- \otimes \sigma_+ - \sigma_+ \otimes \sigma_-}$$

This eqn describes how the coefficient matrix A (from the discretized Poisson eqn) is decomposed using tensor products. The goal is to break down larger matrices (the matrix A) into smaller, simple operators that are easier to implement in quantum computing.
<-> Where quantum circuits can implement these smaller operators directly.

So what does Each term mean?

is to 'break down larger matrices (the matrix A) into smaller, simple operators that are easier to implement in quantum computing.
↳ Where quantum circuits can implement these smaller operators directly.

So, what does Each term mean?

Breaking down each term:

(1) $I \otimes A_{m-1}$ – The Identity Tensor Product
→ I being the Identity matrix (which note: does not change any vector it acts on)
→ A_{m-1} is the coefficient matrix of a smaller system (i.e. one dimension lower)
The tensor product $I \otimes A_{m-1}$ represents applying A_{m-1} in one part of the system while keeping the other part unchanged

Essentially, A is recursively defined, using a smaller version of itself. (recursive form allows solving higher-dimensional Poisson eqns by building on lower-dimension sol^ns).

(2) $-\sigma_- \otimes \sigma_+$ & $-\sigma_+ \otimes \sigma_-$ – Coupling Terms (Represents state transitions)

→ $\sigma_- = |1\rangle\langle 0|$ → This operator removes a quantum excitation (turns a 1 into a 0).
→ $\sigma_+ = |0\rangle\langle 1|$ → This operator adds a quantum excitation (turns a 0 into a 1).
→ The tensor product, $\sigma_- \otimes \sigma_+$, describes hopping or interaction btwn two sites. – neighbouring sites

Why are these terms subtracted?
→ These terms represent nearest-neighbour interactions in a discretized system. In finite-difference methods, the Poisson eqn is approximated using terms like:

$$\boxed{\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}}$$

which involves interactions ω neighbours. The (-)ive sign ensures that these interactions correctly mimic the Laplacian operator.

Quantum Circuit & Measurement

The quantum circuit consists of:
→ A parameter ansatz (Quantum Approximate Optimization Ansatz - QAOA).
→ Observables designed to efficiently compute expectation values

A key feature is using Hadamard tests to measure expectation values of the decomposed matrix elements

The # of measurements scales as: $\boxed{O(\log_2 n+1)}$ → which is significantly fewer than the Pauli basis decomposition method

Experimental Results

The algorithm is tested using PROJECTQ, a quantum computing simulator.

Results:
→ Fidelity (accuracy) improves ω the # of circuit layers
→ A fidelity of 0.99 is achieved ω an optimized # of circuit layers
→ The computational depth scales efficiently for NISQ devices

Key Findings

→ The proposed VQA is efficient for solving the Poisson Eqn on NISQ devices.
→ The explicit decomposition of A reduces the # of quantum measurements.

→ The proposed VQA is efficient for solving the Poisson Eqn on NISQ devices.
→ The explicit decomposition of A reduces the # of quantum measurements.
→ The algorithm can be extended to certain boundary conditions (e.g. Neumann, Robin).
→ The approach can also be used for solving general tridiagonal & pentadiagonal matrices.

→ know b

Solve a 4x4 matrix ($A\vec{x}=\vec{b}$) using Qiskit
(or 8x8)

tridiagonal (given)

find x (exact solⁿ)

(1) pick solⁿ x ⇒ $A\vec{x}=\vec{b}$

(2) Now, use A b to find x in Qiskit
$x_{quantumly}$

(3) Compare X_{act} w/ $X_{quantum}$ & see if they agree