A Hybrid Quantum-Classical Framework for Solving Computational Fluid Dynamics and Nuclear Physics Equations using Quantum Singular Value Decomposition and Transformation

Devon Bontrager[1], Kareem El-Araby[1], Manu Chaudhary[2], Alvir Nobel[1], Pruthviraj Sadhankar[1], Shivansh Shrivas[1], Trey Campbell[1], Jacob Spry[1], Naveed Mahmud[3], Chao Lu[4], Kalyan Gottiparthi[4], Murali Gopalakrishnan Meena[4], Zain Karriem[4], and Esam El-Araby[1]

[1]Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS, USA | Emails: {devonibontrager, kareem_el_araby, islam.alvir, pruthviraj.sadhankar, shivansh, trey.campbell, jacobspry, esam}@ku.edu

[2]Department of Electrical Engineering, Illinois State University, Normal, IL, USA | Email: mchaud4@ilstu.edu

[3]Department of Electrical Engineering and Computer Science, Florida Institute of Technology, Melbourne, FL, USA | Email: nmahmud@fit.edu

[4]Oak Ridge National Laboratory, Oak Ridge, TN, USA | Emails: {luc1, gottiparthik, gopalakrishm, karriemz}@ornl.gov

Introduction and Motivation

- Quantum computing offers the potential for exponential speedup over classical methods, helping address the computational challenges of solving multidimensional partial differential equations (PDE) in computational fluid dynamics (CFD) and nuclear physics.
- Existing quantum linear solver algorithms (QLSAs), such as Harrow-Hassidim-Lloyd (HHL) and variational quantum linear solvers (VQLS), have been applied to problems like Hele-Shaw flow and the neutron transport equation (NTE), but they suffer from scalability, execution time, and accuracy limitations, especially for ill-conditioned systems.
- This work introduces a hybrid quantum-classical framework for solving PDEs based on Quantum Singular Value Decomposition (QSVD) and Quantum Singular Value Transformation (QSVT) to overcome these challenges.
- The proposed algorithms were validated on CFD applications, including potential flow past a 2D cylinder, Joukowsky airfoil, and Jayhawk geometries.
- These methods were also applied to nuclear physics problems, specifically solving the multidimensional Boltzmann Transport Equation (BTE) using the diffusion approach for the Neutron Transport Equation (NTE) in nuclear reactor engineering.
- Performance evaluations on both noise-free and noisy environments of IBM Quantum simulators show that the QSVD and QSVT solvers achieve higher accuracy, better scalability, and faster execution times than existing quantum PDE solvers.
- The study further provides analytical comparisons of the temporal and spatial complexities of the proposed QSVD and QSVT techniques, demonstrating their computational advantages.

Background

Finite Difference Method (FDM)

- Numerical technique to solve PDEs by approximating them with difference equations.
- Converts continuous domain problem into a finite set of grid points.

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

Fundamentals of Quantum Computing

- Quantum computers utilize superposition and entanglement.
- Quantum bit or qubit is the fundamental unit of quantum computation.
- For an n-qubit system, the quantum state vector is represented as shown:

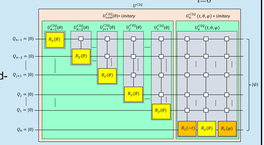
$$|\psi\rangle = \sum_{j=0}^{N-1} c_j \cdot |j\rangle, \text{ where } \sum_{j=0}^{N-1} |c_j|^2 = 1, \text{ and } 0 \le j < N$$

$$|\psi\rangle = U^{C2Q} \cdot |\psi_0\rangle = U^{C2Q} \cdot |0\rangle^{\otimes n} = \sum_{i=0}^{N-1} c_i |i\rangle \quad \text{where} \sum_{i=0}^{N-1} |c_i|^2 = 1$$

- Quantum gates are unitary operations which can change the state of a qubit.

Classical-to-quantum encoding (C2Q)

- C2Q is a depth-optimized amplitude encoding technique [1] to encode classical data into quantum domain.
- Encodes normalized classical data vector into quantum domain.
- Generates shallower circuit by using 50% fewer gates compared to nearest alternative [2].
- Suitable for NISQ devices because shallow circuits are more resistant to decoherence noise.



Singular Value Decomposition (SVD)

$$A = W\Sigma V^{\dagger}$$

- Decompose a matrix into two orthogonal matrices and a diagonal matrix.

Polar Decomposition

$$A = PU$$

- Decompose a matrix into two unitary matrices and a positive semi-definite symmetric matrix.

Column-by-column decomposition (CCD)

- Decomposes any isometry into single qubit and CNOT gates [3].
- Used in ***QuantumCircuit.unitary()*** function of ***Qiskit*** [4].

Projected Unitary Encodings and Block-Encoding

- QSVT represents a generally non-unitary operator **A** through a unitary operator **U** and orthogonal projectors [5], such that $A := \tilde{\Pi} U \Pi$
- By extending the system with ancilla qubits and projecting onto the ancilla zero state, the operator **A** is embedded as the top-left block of a larger unitary matrix **U**, satisfying

$$\tilde{\Pi} = \Pi = |0\rangle\langle 0|^{\otimes a} \otimes I_r \qquad ((\langle 0|^{\otimes a} \otimes I_r)\, U\, (|0\rangle^{\otimes a} \otimes I_r) = A$$

Polynomial Approximation via QSVT

- By alternating the block-encoding unitary its adjoint, and projector-controlled phase gates, QSVT constructs high-degree polynomial transformations of the encoded matrix.

$$\prod_{k=1}^{d/2} \Pi_{\phi_{2k-1}} U(A)^{\dagger} \Pi_{\phi_{2k}} U(A) \Big] \Pi_{\phi_{d+1}} = \begin{pmatrix} P(A) & * \\ * & * \end{pmatrix}$$

Related Work

Performance study of VQAs for solving the Poisson equation on a quantum computer [6]

- Uses variational quantum linear solver (VQLS) for solving Poisson equations
- Poor hardware results, low accuracy and long execution times

Variational quantum algorithm based on the minimum potential energy for solving the Poisson equation [7]

- Solves the Poisson equation by formulating a minimum potential energy-based cost function
- VQE-based approach for solving PDEs but it exhibits low accuracy and long execution times on NISQ devices

Quantum algorithm for systems of linear equations with exponentially improved dependence on precision [8]

- Requires significantly less time and has a better precision than HHL algorithm
- Uses Fourier or Chebyshev series
- Not suitable for NISQ devices because of their high error rates, limited coherence, and complex operations

Hybrid classical-quantum approach to solve heat equation using quantum annealers [9]

- Linear system is encoded as a quadratic unconstrained binary optimization (QUBO) problem
- Solves the steady-state 2D heat equation using D-Wave systems
- Time dependent heat analysis not shown, and the approach suffers from scalability issues

A quantum approach for exploring the numerical results of the heat equation [10]

- Solves 1D heat equation using Trotter Suzuki decomposition
- Suffers from high error rate and limited scalability

Proposed Methodology

Proposed PDE Solver for Multidimensional PDEs

- The proposed PDE solver algorithm offers two methods leveraging QSVD and QSVT respectively.
- Classical pre-processing step rebalances the system numerically using Ruiz Equilibration.

$$Au = b \qquad \tilde{A} = D_r A D_c, \qquad \tilde{b} = D_r b, \qquad u = D_c \tilde{x} \qquad \tilde{A}\tilde{x} = \tilde{b}$$

Method 1 (QSVD)

- Method 1 uses classical-to-quantum (C2Q) encoding, finite difference method (FDM), and column-by-column decomposition (CCD) [4].
- Using FDM, the partial differential equation (PDE) is converted into a system of linear equations.
- SVD and polar decomposition has been used to find the value of P^{-1} and $U^{\dagger}$ as shown below:

$$\tilde{A} = PU = W\Sigma V^{\dagger}, \text{ where } P = W\Sigma W^{\dagger}, U = WV^{\dagger}, P^{-1} = W\Sigma^{-1}W^{\dagger}$$

- System matrix A is preprocessed using polar decomposition, rearranged, and normalized as shown:

$$(PU)\vec{u} = \vec{b} \Rightarrow U\vec{u} = P^{-1}\vec{b} \Rightarrow \vec{u} = U^{\dagger}P^{-1}\vec{b} \Rightarrow \frac{\vec{u}}{\left\|P^{-1}\vec{b}\right\|} = U^{\dagger} \cdot \left(\frac{P^{-1}\vec{b}}{\left\|P^{-1}\vec{b}\right\|}\right)$$

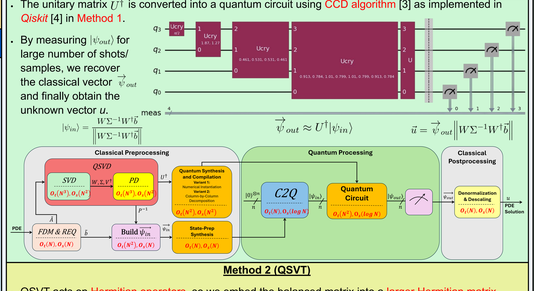
$$|\psi_{out}\rangle = U^{\dagger} \cdot |\psi_{in}\rangle, \text{ where } |\psi_{out}\rangle = \frac{\vec{u}}{\left\|P^{-1}\vec{b}\right\|}, \text{ and } |\psi_{in}\rangle = \frac{P^{-1}\vec{b}}{\left\|P^{-1}\vec{b}\right\|}$$

Circuit complexity with N number of PDE unknowns:

- Classical Temporal: $O(N^3)$
- Classical Spatial: $O(N^2)$
- Quantum Temporal: $O(N^2)$
- Quantum Spatial: $O(\log_2 N)$

- The unitary matrix $U^{\dagger}$ is converted into a quantum circuit using CCD algorithm [3] as implemented in *Qiskit* [4] in Method 1.
- By measuring $|\psi_{out}\rangle$ for large number of shots/samples, we recover the classical vector $\vec{\psi}_{out}$ and finally obtain the unknown vector u.

$$\vec{\psi}_{out} \approx U^{\dagger}|\psi_{in}\rangle \qquad \vec{u} = \vec{\psi}_{out}\left\|W\Sigma^{-1}W^{\dagger}\vec{b}\right\|$$



Method 2 (QSVT)

- QSVT acts on Hermitian operators, so we embed the balanced matrix into a larger Hermitian matrix.

$$H = \begin{bmatrix} 0 & \tilde{A} \\ \tilde{A}^{\dagger} & 0 \end{bmatrix} \qquad \tilde{H} = \frac{H}{\|H\|} \qquad U = \begin{bmatrix} \tilde{H} & \sqrt{I - \tilde{H}^2} \\ \sqrt{I - \tilde{H}^2} & -\tilde{H} \end{bmatrix} \qquad P_d(\tilde{H}) \approx \tilde{H}^{-1} = \|H\| . H^{-1}$$

- We approximate a bounded version of it on the spectral interval.
- Because the operator was embedded into a larger Hermitian matrix, the right-hand side must also be embedded.

$$|b_{emb}\rangle = \begin{bmatrix} |b\rangle \\ 0 \end{bmatrix}$$

- One QSVT step means: apply the block encoding, then apply a phase on the ancilla

$$\Phi(\phi_k) = e^{i\phi_k Z_a} \otimes I^{\otimes(n+1)} = R_z(-2\phi_k) \otimes I^{\otimes(n+1)}, \qquad k = 0, 1, 2, \ldots, d \qquad |\psi_0\rangle = |0\rangle_a \otimes |b_{emb}\rangle$$

- X gates are ancilla basis toggles for the complementary QSVT phase branch:

$$|\psi_1\rangle = \Phi(\phi_0)|\psi_0\rangle$$

$$W_{\vec{\phi}} = U\Phi(\phi_d)U^{\dagger}\Phi(\phi_{d-1})U\Phi(\phi_{d-2})\cdots U^{\dagger}\Phi(\phi_1)U\Phi(\phi_0) \rightarrow d \text{ is even} \qquad |\psi_2\rangle = U|\psi_1\rangle$$

$$W_{\vec{\phi}} = U^{\dagger}\Phi(\phi_d)U\Phi(\phi_{d-1})U^{\dagger}\Phi(\phi_{d-2})\cdots U^{\dagger}\Phi(\phi_1)U\Phi(\phi_0) \rightarrow d \text{ is odd} \qquad P_d(\tilde{H})|b_{emb}\rangle \approx \alpha H^{-1}|b_{emb}\rangle$$

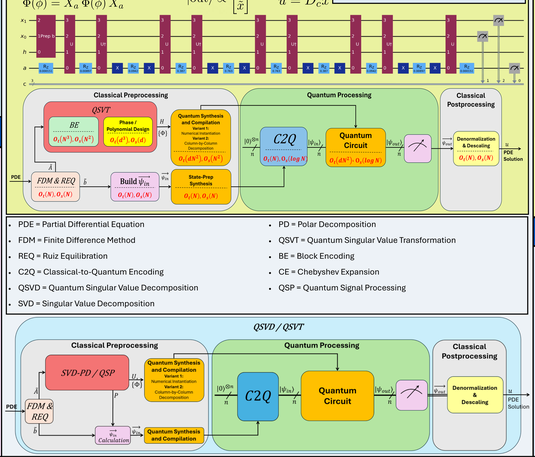
$$(\langle 0|_a \otimes I^{\otimes(n+1)})W_{\vec{\phi}}(|0\rangle_a \otimes I^{\otimes(n+1)}) \approx P_d(\tilde{H})$$

- The optimized QSVT (QSVT-Opt) bypasses this repeated chain of U and $\Phi(\phi_k)$ gate execution by pre-computing the entire product matrix $W_{\vec{\phi}}$ classically and then synthesizing the quantum circuit using a column-by-column (CCD) decomposition.
- Repeating those phase-and-unitary steps builds the full polynomial transformation.

$$\tilde{\Phi}(\phi) = X_a\, \Phi(\phi)\, X_a \qquad |\text{out}\rangle \propto \begin{bmatrix} 0 \\ \tilde{x} \end{bmatrix} \qquad u = D_c\tilde{x}$$

Circuit complexity with N number of PDE unknowns and d degree of polynomial:

- Classical Temporal: $O(N^3 + dN^2 + d^3)$
- Classical Spatial: $O(N^2 + d^2)$
- Quantum Temporal: $O(dN^2)$
- Quantum Spatial: $O(\log_2(N) + 2)$



- PDE = Partial Differential Equation
- FDM = Finite Difference Method
- REQ = Ruiz Equilibration
- C2Q = Classical-to-Quantum Encoding
- QSVD = Quantum Singular Value Decomposition
- SVD = Singular Value Decomposition
- PD = Polar Decomposition
- QSVT = Quantum Singular Value Transformation
- BE = Block Encoding
- CE = Chebyshev Expansion
- QSP = Quantum Signal Processing

Case Study 1: Computational Fluid Dynamics (CFD)

- The fundamental idea is to find a mathematical equation that can predict the behavior of fluid flow.
- Governing equations can be Navier-Stokes equations or a simplified model derived using reasonable assumptions.
- The problem is usually formulated as a Partial Differential Equation (PDE).
- In fluid mechanics we are mainly interested in velocity field. For two-dimensional flow, the velocity has two components: u(x,y), v(x,y) are velocity in the x, y -directions respectively.
- Instead of solving two coupled equations, we can apply a mathematical transformation to simplify the formulation.
- Potential flow is a type of flow in which the velocity field can be expressed as the gradient of a scalar function, called the velocity potential.

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} = 0$$

- We need two conditions:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0 \xrightarrow{\rho \text{ is constant} = \frac{\partial \rho}{\partial t} = 0} \nabla \cdot (\rho \vec{u}) = 0 \xrightarrow{\div \rho} \nabla \cdot (\vec{u}) = 0$$

1) Incompressible (ρ = constant) —> Divergence theorem 2) Irrotational (no rotation/vorticity) —> Stokes theorem

$$\nabla = (\partial/\partial x, \partial/\partial y) = \text{gradient}, \quad \vec{u} = (u_x, u_y) = \text{velocity vector}, \quad \rho = \text{fluid density}$$

$$\partial u_y/\partial x - \partial u_x/\partial y = 0 \rightarrow \partial/\partial x(-\partial\psi/\partial x) - \partial/\partial y(\partial\psi/\partial y) = 0$$

- For incompressible flow, define stream function:

$$\rightarrow -\partial^2\psi/\partial x^2 - \partial^2\psi/\partial y^2 = 0 \rightarrow \nabla^2\psi = \partial^2\psi/\partial x^2 + \partial^2\psi/\partial y^2 = 0 \rightarrow \nabla^2\psi = 0$$

- Discretization: Finite difference method converts PDE to linear system using Taylor series expansion:

$$\frac{\partial^2\psi}{\partial x^2} \approx \frac{\psi(x_{i+1}, y_j) - 2\psi(x_i, y_j) + \psi(x_{i-1}, y_j)}{h^2}$$

$$\frac{\partial^2\psi}{\partial y^2} \approx \frac{\psi(x_i, y_{j+1}) - 2\psi(x_i, y_j) + \psi(x_i, y_{j-1})}{h^2}$$

- A 5-point stencil:

$$\nabla^2\psi_{ij} \approx \frac{\psi_{i-1,j} + \psi_{i+1,j} + \psi_{i,j-1} + \psi_{i,j+1} - 4\psi_{ij}}{h^2} = 0$$

- Cylinder: We evaluate a point in a cylinder using the simple formula $x^2 + y^2 \le R^2$ and use the far-field boundary condition $\psi(x, y) = U_{\infty}.y.\left(1 - \frac{R^2}{r^2}\right)$
- Solving potential flow past a 2D cylinder: $A\psi = \mathbf{b}$

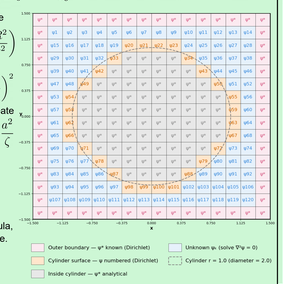
$$|u| = \sqrt{u_x^2 + u_y^2} \quad u_x = \frac{\partial\psi}{\partial y}, \quad u_y = -\frac{\partial\psi}{\partial x} \quad c_p = 1 - \left(\frac{u}{U_{\infty}}\right)^2$$

- Airfoil: We apply complex Joukowsky transformation to create an airfoil shape then a Point in Polygon (PIP) algorithm to check if a point is inside a curved airfoil boundary.

$$Z = \zeta + \frac{a^2}{\zeta}$$

  - ξ: a point on a circle in the complex plane, the input shape before transformation.
  - **a**: a real scaling constant that controls the size and sharpness of the resulting airfoil.
  - **Z**: the output complex number after applying the formula, tracing the airfoil boundary as ζ walks around the circle.
  - Far-field boundary condition:

$$\psi(x, y) = U_{\infty}.(y.\cos\alpha - x.\sin\alpha)$$



Case Study 2: Neutron Transport Equation (NTE)

General BTE:

$$\frac{\partial f}{\partial t} + \frac{\vec{p}}{m} \cdot \nabla_{\vec{r}} f + \vec{F} \cdot \nabla_{\vec{p}} f = \left(\frac{\partial f}{\partial t}\right)_{\text{coll}}$$

$$\left(\frac{\partial f}{\partial t}\right)_{\text{loss}} = -v\Sigma_t f = -\Sigma_t \psi$$

Collisions out term

$$\frac{\vec{p}}{m} \cdot \nabla_{\vec{r}} f \rightarrow \mathbf{v} \cdot \nabla f \rightarrow v(E)\Omega \cdot \nabla\frac{\psi}{\nu} \rightarrow \Omega \cdot \nabla\psi$$

streaming term

$$\int_{4\pi} \Sigma_s \psi(r, \Omega', E')\, d\Omega' dE'$$

Scattering in term

NTE:

$$\frac{1}{v(E)} \frac{\partial\psi}{\partial t} + \Omega \cdot \nabla\psi + \Sigma_t\psi = \int_0^{\infty}\int_{4\pi} \Sigma_s \psi(r, \Omega', E')\, d\Omega' dE' + S$$

$$\frac{\partial f}{\partial t} \rightarrow \frac{1}{v(E)} \frac{\partial\psi}{\partial t}$$

derivative term

Assuming steady-state, $\frac{\partial\psi}{\partial t} \rightarrow 0$

S_n Equation (One Direction):

$$\Omega_n \cdot \nabla\psi_n(r, E) + \Sigma_t(r, E)\psi_n(r, E) = \sum_{m=1}^{N} w_m \int_0^{\infty} \Sigma_s\, \psi_m(r, E')\, dE' + S_n(r)$$

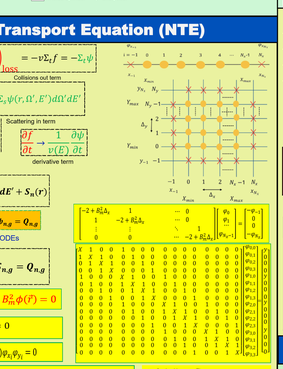
Multi-group Energy Discretization: This is one PDE per discrete direction

$$\frac{d\psi_{n,g}}{ds} + \Sigma_{t,g}\psi_{n,g} = Q_{n,g}$$

First order ODEs

Energy is divided into g groups: $\psi_n(r, E) \rightarrow \psi_{n,g}(r)$

$$\Omega_n \cdot \nabla\psi_{n,g} + \Sigma_{t,g}\psi_{n,g} = \sum_{g'=1}^{G}\sum_{m=1}^{N} w_m\, \Sigma_{s,g'\rightarrow g}\psi_{m,g'} + S_{n,g} = Q_{n,g}$$



Diffusion form	
	$\nabla^2\phi(\vec{r}) + \left[\frac{\nu\Sigma_f - \Sigma_a}{D}\right]\phi(\vec{r}) = 0$ $\quad \nabla^2\phi(\vec{r}) + B_m^2\phi_i(\vec{r}) = 0$
1D NTE	$\varphi_{x_i-1} - 2\varphi_{x_i} + \varphi_{x_i+1} + B_m^2\varphi_{x_i}\Delta_x = 0$
2D NTE	$\varphi_{(i-1,j)} + \varphi_{(i+1,j)} + \varphi_{(i,j-1)} + \varphi_{(i,j+1)} + (-4 + B_m^2\Delta^2)\varphi_{i,j} = 0$
3D NTE	$\varphi_{x_{i-1},y_i,z_i} + \varphi_{x_i+1,y_i,z_i} + \varphi_{x_i,y_{i-1},z_i} + \varphi_{x_i,y_i+1,z_i} + \varphi_{x_i,y_i,z_i-1} + \varphi_{x_i,y_i,z_i+1} + (-6 + B_m^2\Delta^2)\varphi_{x_i,y_i,z_i} = 0$

$X = -4 + B_m^2\Delta^2$ $\quad y = -\varphi_B$ $\quad \Delta = \min(\Delta_x, \Delta_y)$

$\Delta_x = \frac{X_{max} - X_{min}}{N_x - 1}$ $\quad \Delta_y = \frac{Y_{max} - Y_{min}}{N_y - 1}$ $\quad \Delta_z = \frac{z_{max} - z_{min}}{N_z - 1}$

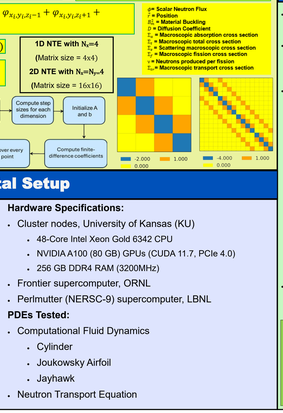
1D NTE with N_x=4 (Matrix size = 4x4)

2D NTE with N_x=N_y=4 (Matrix size = 16x16)

Assumptions for NTE in Diffusion form:

- Monoenergetic
- Linearly anisotropic
- Time independent
- No independent source

- ξ^{α} Scalar Neutron Flux
- Φ_i = Fission
- B_m^2 = Material Buckling
- D = Diffusion Coefficient
- Σ_a = Macroscopic absorption cross section
- Σ_f = Macroscopic total cross section
- Σ_s = Scattering macroscopic cross section
- Σ_t = Macroscopic fission cross section
- ν = Neutrons produced per fission
- Σ_t = Macroscopic transport cross section



Experimental Setup

Metrics:

- Scalability quantified in terms of number of datapoints
- Fidelity
- Total execution time
- Circuit Depth
- Gate count

Simulators and Emulators:

- Noise-free (statevector simulator)
- Noisy (AerSimulator)
- Hardware Emulator (FakeNighthawk)
- Grid size: 16x16
- No. of Qubits: 8 (QSVD), 10 (QSVT)
- Circuit samples (shots): 32M

Hardware Specifications:

- Cluster nodes, University of Kansas (KU)
  - 48-Core Intel Xeon Gold 6342 CPU
  - NVIDIA A100 (80 GB) GPUs (CUDA 11.7, PCIe 4.0)
  - 256 GB DDR4 RAM (3200MHz)
- Frontier supercomputer, ORNL
- Perlmutter (NERSC-9) supercomputer, LBNL

PDEs Tested:

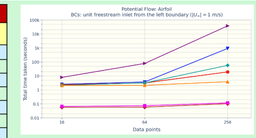
- Computational Fluid Dynamics
  - Cylinder
  - Joukowsky Airfoil
  - Jayhawk
- Neutron Transport Equation

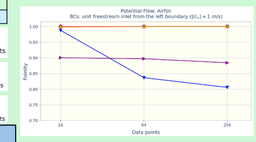
Experimental Results

Computational Fluid Dynamics (CFD)

- Grid size: 16x16
- No. of Qubits: 8 (QSVD), 10 (QSVT)
- Shots: 32M
- Free-stream velocity
  - U_{∞}= 1 m/s
- Cylinder
  - Symmetric flow
    - No lift (Cl=0)
- Joukowsky Airfoil
  - Asymmetric pressure (C_p>0 below, C_p<0 above)
    - Generates lift →$C_L \approx 2\pi\alpha \approx 1.316$

Object	Solver	Fidelity			Gate Count		Circuit Depth	
		State Vector	AerSim	Emulator	AerSim	Emulator	AerSim	Emulator
Cylinder	QSVD	1.0	0.999999	0.808936	174	327,973	161	195,892
	QSVT	0.999997	0.999962	N/A	1,202	4,825,710,458	1,201	2,974,079,518
	QSVT-Opt	0.999997	0.999952	0.836766	1,416,904	14,831,400	1,416,903	9,205,237
Airfoil	QSVD	1.0	0.999999	0.805838	263	330,933	250	197,763
	QSVT	0.999992	0.999953	N/A	1,202	4,817,703,210	1,201	2,950,608,624
	QSVT-Opt	0.999992	0.999954	0.884381	1,416,904	14,849,910	1,416,903	9,224,946
Jayhawk	QSVD	1.0	0.999999	0.671472	263	328,263	250	195,248
	QSVT	0.999997	0.999942	N/A	1,202	4,818,219,250	1,201	2,950,538,026
	QSVT-Opt	0.999997	0.999950	0.852658	1,416,904	14,852,044	1,416,903	9,228,859

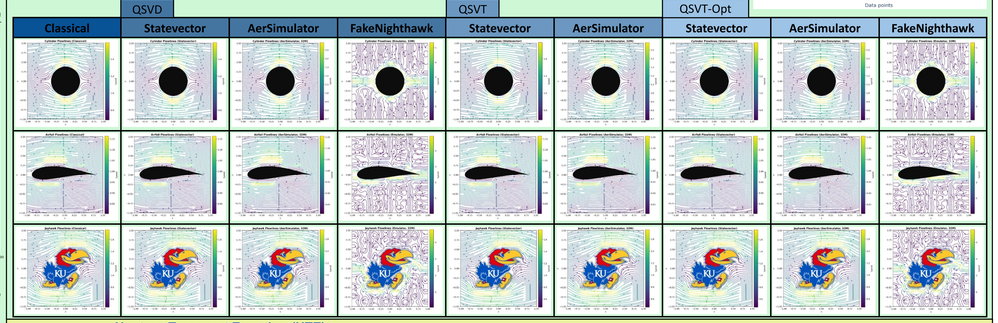




Simulation Environments

a. Classical
b. QSVD (Statevector Simulator)
c. QSVD (AerSimulator)
d. QSVD (Hardware Emulator)
e. QSVT (Statevector Simulator)
f. QSVT (AerSimulator)
g. QSVT-Opt (Statevector Simulator)
h. QSVT-Opt (Hardware Emulator)
i. QSVT-Opt (Hardware Emulator)

- QSVD-Qiskit (statevector)
- QSVD-Qiskit (AerSim), 32M shots
- QSVD-Qiskit (Emulator), 32M shots
- QSVT-Qiskit (statevector)
- QSVT-Qiskit (AerSim), 32M shots
- QSVT-Qiskit (Emulator), 32M shots
- QSVT-Opt-Qiskit (statevector)
- QSVT-Opt-Qiskit (AerSim), 32M shots
- QSVT-Opt-Qiskit (Emulator), 32M shots



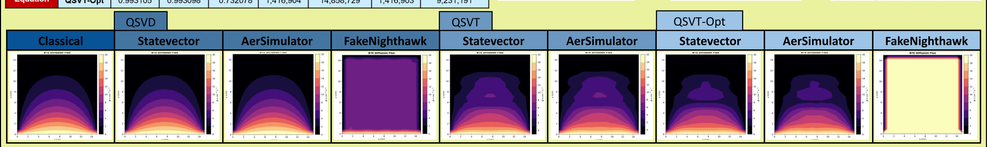
Neutron Transport Equation (NTE)

- Grid size: 16x16
- No. of Qubits: 8 (QSVD), 10 (QSVT)
- Shots: 32M
- 2D neutron transport equation
- X_{diff} = 15 cm, Y_{diff} = 15 cm
- Material buckling B^2_m for U_{238} as −3.9615×10⁻⁵ cm⁻²
- One boundary value (BV) is ϕ(x = 0) = 20 n cm⁻² s⁻¹ and the remaining BVs are 0

	Solver	Fidelity			Gate Count		Circuit Depth	
		State Vector	AerSim	Emulator	AerSim	Emulator	AerSim	Emulator
2D Neutron Transport Equation	QSVD	1.0	0.999999	0.735440	394	171,442	380	105,288
	QSVT	0.993105	0.992948	N/A	1,202	4,796,216,506	1,201	2,935,190,088
	QSVT-Opt	0.993105	0.993098	0.732078	1,416,904	14,858,729	1,416,903	9,231,191



- QSVD-Qiskit (statevector)
- QSVD-Qiskit (AerSim), 32M shots
- QSVD-Qiskit (Emulator), 32M shots
- QSVT-Qiskit (statevector)
- QSVT-Qiskit (AerSim), 32M shots
- QSVT-Qiskit (Emulator), 32M shots
- QSVT-Opt-Qiskit (statevector)
- QSVT-Opt-Qiskit (AerSim), 32M shots
- QSVT-Opt-Qiskit (Emulator), 32M shots



Conclusions and Future Work

- This work presents a hybrid quantum-classical framework and complete methodology for solving PDEs using QSVD- and QSVT-based solvers.
- Experimental results from real-world use cases in computational fluid dynamics and nuclear physics show that the methodology is feasible and produces solutions of high fidelity in noisy simulation environments.
- We have also presented critical metrics such as fidelity, execution time, realistic gate counts, circuit depth from our quantum hardware emulation runs.
  - Both methods achieve near-unit fidelity in the ideal statevector and shot-based AerSim environments (QSVD ≈ 1.0, QSVT and QSVT-Opt ≈ 0.993), confirming that all solvers implement the target solution correctly at the algorithmic level.
  - Under realistic (FakeNighthawk) noise, QSVD fidelity degrades with problem size, falling from ≈ 0.85 to ≈ 0.73 beyond a few hundred data points, reflecting accumulated gate noise over deeper circuits.
  - On the emulator, QSVD remains tractable (~328k gates, ~195k depth-layers), whereas native-gate decomposition inflates QSVT to ~4.8 billion gates and ~2.9 billion depth-layers, roughly four orders of magnitude more, making its emulation infeasible. The QSVT-Opt decomposition cuts this to ~14.8M gates and ~9.2M depth-layers (roughly two orders of magnitude fewer than QSVT) and recovers emulator fidelity comparable to QSVD (~0.85 for the CFD cases, and ~0.73 for NTE).
- Future work will involve QSVT circuit optimizations for efficient deployment on quantum hardware.

Acknowledgements

This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. This research also used resources of the National Energy Research Scientific Computing Center (NERSC), a Department of Energy Office of Science User Facility under Contract DE-AC02-05CH11231.

References

[1] El-Araby, E., Mahmud, N., Jeng, M.J., MacGillivray, A., Chaudhary, M., Nobel, M.A.I., Islam, S.I.U., Levy, D., Kneidel, D., Watson, M.R. and Bauer, J.G., 2023. Towards complete and scalable emulation of quantum algorithms on high-performance reconfigurable computers. *IEEE Transactions on Computers*, *72*(8), pp.2350-2364.

[2] Shende, Vivek V., Stephen S. Bullock, and Igor L. Markov. "Synthesis of quantum logic circuits." In Proceedings of the 2005 Asia and South Pacific Design Automation Conference, pp. 272-275. 2005.

[3] Iten, Raban, Roger Colbeck, Ivan Kukuljan, Jonathan Home, and Matthias Christandl. "Quantum circuits for isometries." *Physical Review A* 93, no. 3 (2016): 032318.

[4] IBM Quantum. Qiskit: An Open-Source Framework for Quantum Computing. Available online: https://www.ibm.com/quantum/qiskit (accessed on 28 February 2025).

[5] Gilyén, A., Su, Y., Low, G.H. and Wiebe, N., 2019, June. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In Proceedings of the 51st annual ACM SIGACT symposium on theory of computing (pp. 193-204).

[6] Ali, Mazen, and Matthias Kabel. "Performance study of variational quantum algorithms for solving the Poisson equation on a quantum computer." *Physical Review Applied* 20.1 (2023): 014054.

[7] Sato, Y., Kondo, R., Koide, S., Takamatsu, H. & Imoto, N. (2021). Variational quantum algorithm based on the minimum potential energy for solving the Poisson equation. *Physical Review A*, *104*(5), 052409.

[8] Childs, Andrew M., Robin Kothari, and Rolando D. Somma. "Quantum algorithm for systems of linear equations with exponentially improved dependence on precision." *SIAM Journal on Computing* 46.6 (2017): 1920-1950.

[9] Pollachini, G. G., Salazar, J. P., Góes, C. B., Maciel, T. O. & Duzzioni, E. I. Hybrid classical-quantum approach to solve the heat equation using quantum annealers. *Phys. Rev. A* 104, 032426 (2021).

[10] Daribayev, B., Mukhanbet, A., Azatbekuly, N. & Imankulov, T. A quantum approach for exploring the numerical results of the heat equation. *Algorithms* 17, 327 (2024).

[11] Tavella, D., & Randall, C. Pricing Financial Instruments: The Finite Difference Method. *Wiley* (2000).

[12] PennyLane Development Team (2024) *Intro to QSVT: Transforming matrices encoded in matrices*. Available at: https://pennylane.ai/qml/demos/tutorial_intro_qsvt (Last accessed: 21 June 2026).