Quantum CFD in 2025: An Industrial Perspective with Use Cases

Andy Gallo, Mohammad Alhawwary
GE Aerospace Research

Abstract—We provide an industrial perspective on quantum computing in 2025, focusing on applications in computational fluid dynamics (CFD). We examine current quantum algorithm developments for CFD and exercise one, the Harrow–Hassidim–Lloyd algorithm, in benchmark scenarios for the Hele-Shaw flow and 1D supersonic divergent nozzle problems. Based on this we propose a benchmarking approach to be used in tracking quantum computing progress relative to CFD applications. We examine the current vendor roadmaps, as they are, and from there project the timing of quantum utility for industrial applications of CFD.

1 Overview

This report examines the state of quantum computing in 2025 from an industrial perspective, with particular emphasis on applications in computational fluid dynamics (CFD). As we stand at a critical juncture in the development of quantum computing technology, this work synthesizes our hands-on research experience, analysis of the evolving hardware and software landscape, and strategic thinking about when and how quantum computing might deliver industrial utility for CFD applications.

Quantum computing represents a paradigm shift in computation comparable to the transition from analog to digital computing in the 1950s and 60s—an era in which GE was a major innovator in hardware and operating systems. Just as GE Research contributed to early computing through projects like GECOS and later through algorithmic complexity theory via the Turing Award-winning work of Hartmanis and Stearns, we find ourselves again at a moment where understanding emerging computational paradigms is essential to maintaining technological leadership. Today quantum computing has the potential to directly impact multiple aspects of our business: materials science through native quantum system modeling, CFD and optimization through potential solutions to computationally intractable problems, and indirectly through cryptography and networking.

Our investigation centers on a detailed case study of the Harrow–Hassidim–Lloyd (HHL) algorithm applied to CFD problems. We implemented and evaluated HHL on two test cases: the Hele-Shaw flow problem, and a 1D supersonic divergent nozzle using the Euler equations. This work involved modernizing existing quantum linear solver codes, enhancing them for the 1D nozzle case, and developing workflow tools to manage the complexity of quantum-classical hybrid computations. The results illuminate both the promise and current limitations of quantum approaches to CFD.

The HHL algorithm theoretically offers exponential speedup for solving linear systems of the form $Ax = b$, promising $O(\log N)$ complexity versus classical $O(N^3)$ for Gaussian elimination, or polynomial complexity for practical iterative methods. However, our study confirms what theory suggests: extracting useful CFD solutions from quantum states requires measurement-based approaches that scale back to polynomial complexity. When accounting for quantum state preparation overhead and the reality that we need actual solution vectors rather than summary statistics, any advantage is obviated. This does not invalidate the quantum approach — it contextualizes where we are in its development trajectory.

This reality leads us to propose a novel dynamic benchmarking framework for quantum CFD. Rather than competing on execution speed — a contest classical methods will win for the foreseeable future — we focus on accuracy and problem size relative to our continuously advancing classical state-of-the-art. We propose a practical threshold: when quantum CFD algorithms can address 20% of our current problem size with 80% of our desired accuracy, industrial utility has come into view. This benchmark should be re-evaluated periodically as both quantum hardware and classical methods continue to improve, with the quantum team selecting appropriate test cases and methodologies for each assessment cycle.

1.1 About this Report

Our current focus on materials science and CFD is opportunistic and strategic. Materials science benefits from quantum computing's native ability to model quantum mechanical systems, making it a nearer-term application domain. CFD, while more distant in terms of practical utility, represents a core competency critical to our business. Understanding how quantum methods might eventually transform CFD workflows — even if that transformation is years away — positions us to make informed decisions about technology adoption and integration for a time when utility thresholds are crossed.

This report synthesizes our hands-on research experience, critical analysis of the quantum computing landscape, and strategic thinking about paths to industrial utility. It is structured as follows:

Section 2 presents a detailed case study of the Harrow–Hassidim–Lloyd (HHL) algorithm applied to two CFD

problems: the Hele-Shaw flow and a 1D supersonic divergent nozzle. We discuss the algorithm's theoretical advantages, practical limitations, and error profiles.

Section 3 analyzes key learnings from our experimentation with HHL, focusing on error sources, mitigation strategies, and our implementation experience modernizing quantum linear solver codes for current software stacks. Our learnings are presented from the perspective of both CFD domain experts and software practitioners.

Section 4 proposes a dynamic benchmarking framework for evaluating quantum CFD methods. Rather than competing on execution speed, we focus on accuracy and problem size relative to continuously advancing classical methods, proposing an 80/20 utility threshold as a practical indicator of industrial relevance.

2 The HHL Algorithm for CFD - A Study

We present a case study of the Harrow–Hassidim–Lloyd (HHL) algorithm, a quantum algorithm that solves, with some notable caveats, systems of linear equations, $Ax = b$. It is notable for potentially offering an exponential speedup over classical algorithms under certain conditions [1].

While classical CFD computing techniques use algorithms which are of order $O(n^m \log(n))$ or $O(n^m)$, for problem size n and constant m, quantum computing has the potential for $O(\log n)$. The purpose of this study is to qualify this advantage in practical terms.

2.1 Overview, Comparison to Classical Solvers

The traditional classical algorithm which produces the actual solution vector $|x\rangle$ is Gaussian elimination, which runs in $O(N^3)$ time. Practical classical CFD solvers use iterative methods that are faster than Gaussian elimination, but still polynomial in N, and are themselves dependent for performance on the condition number κ - the high κ typical of CFD then requires the use of matrix preconditioning to obtain practical performance.

HHL does not return the solution vector x, but rather presents a quantum state $|x\rangle$ from which the solution must be extracted in post-processing. This workflow pattern is typical of quantum algorithms.

To obtain the promised efficiency from HHL, the matrix A must be sparse and have a small condition number κ. In addition, the user must be satisfied with summary information about the solution, not any full state tomography, not the actual solution vector x. The norm $\|x\|$ is obtainable, and the angle θ between $|x\rangle$ and $|b\rangle$ is also observable. For an $N \times N$ matrix these outcomes are computable in $O(\log N)$ time.

Towards a more direct comparison to classical, the HHL algorithm can be modified to produce the actual solution vector $|x\rangle$ by using a measurement based approach to the quantum circuit implementation. Given the size of $|x\rangle$, the resulting required number of qubits, and the number of shots required to obtain a measurement-based solution, to effectively sample the N-dimensional solution space is costly. For example, for accuracy ϵ and confidence δ, the algorithm requires $O\left(\frac{N}{\epsilon^2} \log \frac{1}{\delta}\right)$ shots [2].

2.2 Description of the HHL Algorithm

HHL starts with the classical vector b, normalizes it, and encodes it as a quantum state $|b\rangle$. This becomes the initial state. The matrix A is encoded for Hamiltonian simulation by a circuit which executes Quantum Phase Estimation (QPE) to extract eigenvalues, followed by a controlled rotational reciprocal calculation, and finally inverse QPE to disentangle the solution state $|x\rangle$, which is then classically post-processed to yield the solution vector x.

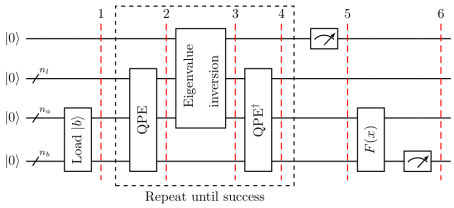


Fig. 1. The modified HHL algorithm. Vector b is encoded as a quantum state $|b\rangle$ and probed by Quantum Phase Estimation (QPE). Measurements across all qubits yield the solution state $|x\rangle$ after post-processing [3].

Each circuit execution produces a bitstring representing the state of all measured qubits. The solution is reconstructed by accumulating the probability distribution over the solution register:

```
For each (bitstring, count):
  state_index = int(bitstring)
  solution_bits = state_index & mask
  quantum_solution[solution_bits] +=
    count/shots
Filter, normalize, scale by ||b||
```

This approach reconstructs the probability amplitudes from the measurement statistics, requiring $O(N)$ to $O(N^2)$ shots for accurate reconstruction of an N-dimensional solution vector. The final scaling by $\|b\|$ restores the solution to the correct magnitude, since HHL operates on the normalized input $|b\rangle = b/\|b\|$. Its possible to modify the weighted averaging which is being used to produce the quantum solution.

The core of HHL is QPE, which works by preparing register qubits in superposition via Hadamard gates, applying controlled unitary operations that encode phase information into the register through phase kickback, then using an inverse quantum Fourier transform to convert the phase into a measurable bitstring representing the eigenvalue. The sequence of unitary operators U^t are a heavy portion of the circuit, are the primary source of error in the algorithm, and are the subject of ongoing research to improve [4].

2.3 Case Study Implementation Details

In Winter 2025 Oak Ridge National Lab conducted an internal training session which included an exercise of the HHL algorithm. This "winter challenge" code tried to leverage ORNL's internal IBM, IonQ, and IQM machines, and in doing so found themselves pinned to a vendor's dependency on an older version of the Qiskit circuit library, the primary

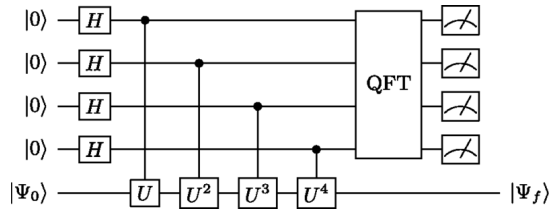


Fig. 2. Quantum Phase Estimation (QPE) circuit

circuit library today [5]. The quantum linear solver library which is used to construct the circuit, transpile it to a specific quantum backend, and execute it on the quantum machine or simulator was two major releases behind. Besides the loss of a number of features and bug fixes, there was also a known performance improvement with the newer versions.

We have a strong preference to stay on the tip of the major library version trees like Qiskit, thus we forked the quantum linear solver code and ported it to the latest Qiskit quantum circuit library [6], [7]. ORNL received the fork back in a pull request and continued to build upon it for further training sessions [8].

In order to orchestrate workflows and keep artifacts separated, we use workflow tooling we developed [9] which is inspired by our industrial experience [10]. The benefits included: sandboxing of library dependencies between different steps in a workflow, tracking of the control and data flow, and the ability to run the quantum circuit on a simulator or quantum machine with a text switch. A file input was added to give easy case construction and parameter sweeps.

These codes where used in the first part of this study - to evaluate the HHL algorithm on a simplified Hele-Shaw steady flow problem. In the second part of the study, we evaluated the HHL algorithm on a more complex problem, a 1D supersonic divergent nozzle with the Euler equations.

Additional codes were developed to perform the HHL algorithm with converging time iterations, more akin to a traditional CFD solver than the original ORNL example [11].

The solver employs a two-level iteration structure:

```
For n = 1 to max_iters (outer):
  For k = 1 to max_inner_iters (inner):
    Compute Jacobian A
    Form RHS b from unsteady residual
    Solve Ax = dQ via HHL:
      Pad to power-of-2, make Hermitian
      Generate/execute quantum circuit
      Extract solution from measurements
    Update q = q + dQ
    If converged: break
  Check outer convergence
```

At the center of the inner loop is the HHL algorithm. Here we're using the same quantum linear solver implementation as in the original ORNL work, but with our own pre-processing and time-stepping workflow. Execution continues to be on the IBM Qiskit simulators. The Frontier HPC machine at Oak Ridge was used for some of the runs, permitting some simultaneous studies, although we discuss elsewhere in this document the limitations of parallelism in the algorithm.

2.4 Numerical Processing

For HHL, as an algorithm for solving $Ax = b$ systems, the matrix A should possess some properties. It has to be a Hermitian matrix with a relatively low condition number κ, ideally not larger than O(1e3). This due to the fact the HHL complexity depends on the sparsity and condition number of the matrix. In industrial CFD however, the matrix A is in general a sparse non-Hermitian matrix with usually large size $N \times N$ and high condition number κ, depending on the mesh size, geometry, boundary conditions, etc.

Matrix A must also be positive definite, which helps ensure numerical stability and convergence. This property is not always guaranteed for arbitrary matrices in real-world CFD scenarios. A variety of techniques can be used to handle a matrix which does not conform to this requirement, at additional computational cost. Table 1 summarizes the matrix size transformations and qubit requirements for HHL.

TABLE 1
Matrix $N \times N$ size transformations and qubit requirements for the HHL algorithm. These are very small matrices relative to current industrial utility. When A is padded, the $|x\rangle$ and $|b\rangle$ vectors are similarly resized to maintain the same dimensionality.

Orig	Padded	Hermitian	$\|x\rangle$ qb	HHL qb
3×3	4×4	8×8	3	$\sim$9
5×5	8×8	16×16	4	$\sim$11
10×10	16×16	32×32	5	$\sim$13
12×12	16×16	32×32	5	$\sim$13
20×20	32×32	64×64	6	$\sim$15
32×32	32×32	64×64	6	$\sim$15

$|x\rangle$ qb = $\log_2(N)$ qubits for state register
HHL qb $\approx 2\times|x\rangle$ qb + 3 (eigenvalue + ancilla + workspace)

Using various preconditioning techniques, the condition number κ can be reduced. This is useful for controlling the depth of the circuit which acts on the qubits. Besides being difficult to execute cleanly on NISQ hardware, and quickly prohibitive on simulators, a deep circuit is costly to construct. Most of the construction time is spent in the QPE and inverse QPE about evenly, with the remainder taking only 1% of the build time.

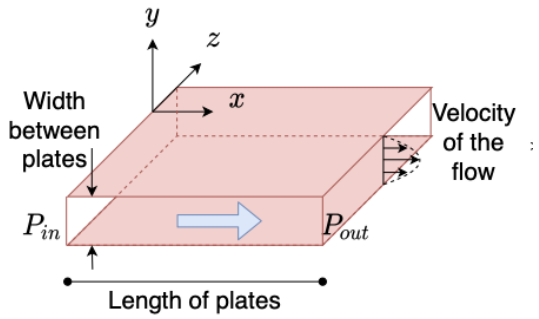
Spectral radius scaling can also be used, where both A and b are divided by the spectral radius ρ (the largest eigenvalue magnitude). While this normalization does not change the condition number κ (since all eigenvalues are scaled uniformly), it does improve numerical stability and helps control circuit depth by constraining eigenvalues to the range $[0, 1]$, which reduces the complexity of the QPE circuit.

2.5 Case Study (1): Hele-Shaw Flow

Two primary cases were executed: the Hele-Shaw flow problem and the 1D supersonic divergent nozzle problem.

In the first case, the Hele-Shaw problem involves modeling fluid flow between two parallel plates separated by a small gap, discretized into a system of linear equations $Ax = b$.

Given some parameters about the geometry, initial conditions, and mesh size, the $Ax = b$ system of equations is discretized and a circuit expressing the HHL algorithm is



$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} = 0 \qquad \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = 0$$

$$\frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_x}{\partial y^2} - \frac{\partial p}{\partial x} = 0 \qquad \frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_x}{\partial y^2} = \frac{\partial p}{\partial x}$$

$$\frac{\partial^2 u_y}{\partial x^2} + \frac{\partial^2 u_y}{\partial y^2} - \frac{\partial p}{\partial y} = 0 \qquad \frac{\partial^2 u_y}{\partial x^2} + \frac{\partial^2 u_y}{\partial y^2} = \frac{\partial p}{\partial y}.$$

$$p = \frac{(P_{in} - P_{out})}{L} x + P_{in}$$

$$u_x = \frac{1}{2\mu} \frac{(P_{out} - P_{in})}{L} y\,(D - y)$$

$$u_y = 0.$$

Fig. 3. Hele-Shaw flow visualization and governing equations. The 2D Hele–Shaw steady flow equations are obtained by imposing incompressibility and inviscid approximations to the Navier–Stokes equations and solving analytically to obtain expressions for pressure and velocity in the x and y directions. [12]

constructed. The b vector is encoded and prepared as $|b\rangle$, and A is used to construct the unitary operations e^{iAt}. The circuit is transpiled to a specific quantum backend and executed on the quantum machine or simulator and x extracted from the quantum result in post-processing. A variety of runs were executed modifying the above parameters as well as the number of qubits used to assess scalability and error characteristics.

2.5.1 Results (1)

As explained above 2.2, we can generate circuits for Hele-Shaw geometry, transpile for a specific quantum backend (real or simulated), execute some number of trial shots on the circuit measuring the outcomes, and post-process the results into a solution vector x. Given the small size of the problems, fidelity is measured relative to a classical solution for the same $Ax = b$ system.

We conducted a series of experiments to explore fidelity and uncertainty quantification. In the simplest case, a 2×2 grid is converted to a circuit and run on a simulator set to a noise model similar to an IBM Heron processor, of the kind available on the IBM cloud today. Parameter sweeps were executed on: varied simulator backends with and without noise models, real quantum computers in the IBM cloud, number of shots, and number of grid points up to 5×5. Multiple runs with the same parameter set were conducted to assess impact on uncertainty quantification.

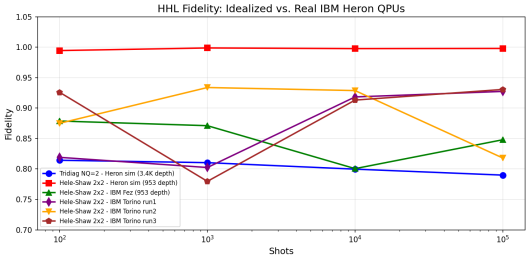


Fig. 4. 2×2 Hele-Shaw and simplified tridiagonal system fidelity comparison, by shots, on low-noise IBM Heron-class simulator and real IBM Heron QPUs Fez (r2) and Torino (r1).

The 2×2 grid results are summarized in Figure 4. For the case on Torino we see fidelity results compared to classical which are encouraging. However, a 2×2 grid is still a very small problem, and the true challenge lies in scaling to larger systems where circuit depth becomes prohibitive. There is also significant uncertainty around even the 2×2 Hele-Shaw results on Torino, which highlights the need for improved error mitigation techniques to achieve reliable solutions at scale.

After multiple runs we notice a trend of improved consistency of results around 10k shots, on this circuit, on this hardware with its noise profile. Below 10k shots we experience insufficient sampling to get stable statistics over the noise. Above 10k shots we start to see diminished returns from long runs. The optimal shot count can be estimated from circuit parameters. For a circuit with depth d and average gate error rate ϵ_g, the circuit fidelity is approximately $F \approx (1 - \epsilon_g)^d$. To overcome the accumulated error and achieve target precision ϵ_p on measurement probabilities, the required shot count is:

$$N_{\text{shots}} \approx \frac{1}{F^2 \epsilon_p^2} = \frac{1}{(1 - \epsilon_g)^{2d} \epsilon_p^2} \tag{1}$$

For the 2×2 Hele-Shaw circuit on Torino with $d \approx 500$, $\epsilon_g \approx 0.002$, and target precision $\epsilon_p \approx 0.05$, this yields $N_{\text{shots}} \approx 10$k, consistent with our empirical observations.

Notice the difference in performance of the same circuit on two other backends - a density matrix simulator fitted with a noise profile of a generic IBM Heron quantum processor, and one with a noise profile of IBM Torino. We see that the experience of running on a simulator does not necessarily transfer directly to a real machine.

The depth differences in the circuits as the mesh size increases highlights a critical challenge: on non-idealized hardware with NISQ-era noise profiles, circuit depth significantly impacts fidelity. Deeper circuits are more susceptible to noise and decoherence, reducing solution accuracy. This is particularly important for HHL, where the quantum phase estimation component becomes deep for larger systems. For the 2×2 Hele-Shaw case, the shallow circuit depth makes it more robust to noise. We have also experimented with a simpler tridiagonal matrix system as a benchmark to check on circuit parameters, and even for this case we have seen the impact of κ on depth, emphasizing the need for error mitigation techniques to maintain scalability even in the simplest cases.

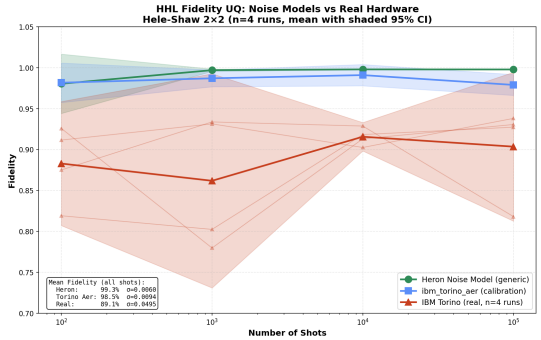


Fig. 5. Uncertainty quantification for 2×2 Hele-Shaw on Torino across 4 runs.

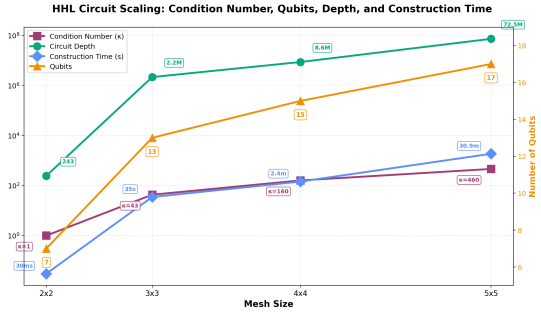


Fig. 6. Key metrics comparison, showing the prohibitive circuit depth and circuit construction costs directly related to κ, as mesh size increases in the Hele-Shaw case. On real quantum hardware execution of deep circuits is limited by coherence times and gate fidelities. On simulators, runtimes quickly become untenable.

2.6 Case Study (2): 1D Supersonic Divergent Nozzle

In the second case, a 1D supersonic divergent nozzle problem was expressed with an Euler system of equations in conservation law form:

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} - G = 0 \tag{2}$$

where:

$$Q = (\rho, \rho u, \rho E)^T \tag{3}$$

$$F = (\rho u, \rho u^2 + p, \rho u H)^T \tag{4}$$

$$G = -\frac{A}{S}\left(\frac{dA}{dx}\right)[\rho u, \rho u^2, \rho u H]^T \tag{5}$$

Here Q is the vector of conserved variables (density, momentum, total energy), F is the flux vector, and G represents source terms arising from the nozzle area variation.

The spatial discretization uses a finite volume method (FVM) with a second-order MUSCL reconstruction scheme and Rusanov flux for the inviscid fluxes. The BDF method is used to discretize the temporal term:

$$\frac{dQ}{dt} = R(Q) \tag{6}$$

$$\left(\frac{I}{\Delta t} - \frac{\partial R}{\partial Q}\right)\Delta Q = R(Q) \tag{7}$$

$$R(Q) = -\left(\frac{1}{A_x}\right)\left(F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}\right) + G \tag{8}$$

The time derivative relation $\frac{dQ}{dt} = R(Q)$ expresses the evolution of the conserved variables. The BDF discretization with the Jacobian term $\frac{\partial R}{\partial Q}$ forms the linear system to be solved at each time step. The residual formulation $R(Q)$ incorporates the finite volume flux differences evaluated at cell interfaces ($i + \frac{1}{2}$ and $i - \frac{1}{2}$). A Newton iteration is used to converge the system towards the final steady solution. The resulting linear system of equations is solved at each iteration using any linear system solver such as LU or GMRES on classical machines.

This is an extension of the original Hele-Shaw case which only solved a single steady flow without time integration. Additionally we are now solving a partial differential system of equations (for ρ (density), ρu (momentum), ρE (energy)) and they are coupled. We also have nonlinear terms whereas the Hele-Shaw case was reduced to only linear terms to produce uncoupled equations for pressure, v_x, and v_y.

Finally, we modified our code and initial conditions for the Euler 1D problem to be able to test the approach of alternating between HHL and the classical CFD cycle to assess the error propagation experimentally.

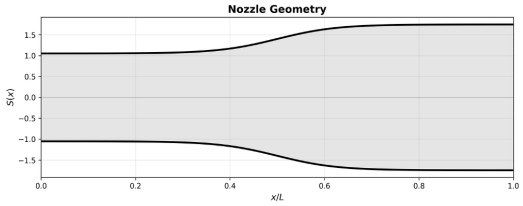


Fig. 7. 1D supersonic nozzle: geometry

2.6.1 *Results (2)*

Here as in the Hele-Shaw case, we can generate circuits for the 1D nozzle geometry using the HHL algorithm, transpile for a specific quantum backend (real or simulated), execute some number of trial shots on the circuit measuring the outcomes, and post-process the results into a solution vector x. But unlike the Hele-Shaw case, we are now iterating over time steps to converge to a steady solution.

Again we sweep over varied simulator backends, vary the number of shots beyond 100k, and number of grid points up to 5×5. The number of iterations allowed for convergence can also be modified to see the impact of longer runs. Multiple runs with the same parameter set were conducted to assess impact on uncertainty quantification.

On an idealized simulator, in the absence of noise models, we can see that HHL solution matches exactly the classical solution. However, simulation time is much longer than classical - the times to generate and execute circuits and post-process results is on order the same as the Hele-Shaw case above for the same sized A matrix and κ. Additionally, our 1D nozzle case iterated over time, thus the total runtime is significantly higher due to the iterative nature of the process, and not parallelizable across time steps. Because of these heavy computational costs, we were limited to smaller mesh sizes up to 5×5. As with Hele-Shaw, the HHL algorithm will require us to grow the size of A to make it

sized a power of 2 and be Hermitian. For example, for a 5 element mesh and three variables A is originally 15×15, but is padded to 16×16 to be a power of 2, and then made Hermitian raising it to 32×32.

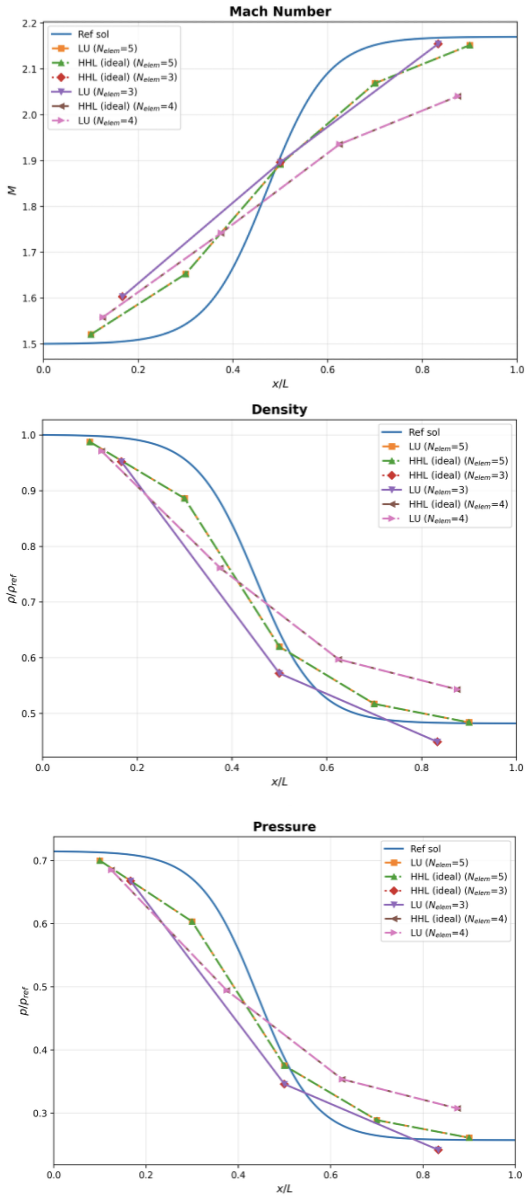


Fig. 8. 1D nozzle: Mach number, density, and pressure distributions from HHL runs on idealized simulator vs classical. The number of elements of the mesh are varied from 3×3 up to 5×5.

The residuals from each iteration are noted and the next iteration is initialized with the previous solution as the starting point. This iterative process continues until the residual falls below a specified tolerance or the maximum specified iterations have been performed.

The cost of each iteration can vary, scaling with the condition number κ, which can vary. Figure 6 above shows the depth and construction time of circuits as κ varies.

Multiple runs with different random seeds show consistent convergence behavior, indicating the method's robustness to initialization. Here we see that increasing the number of shots per iteration generally improves convergence

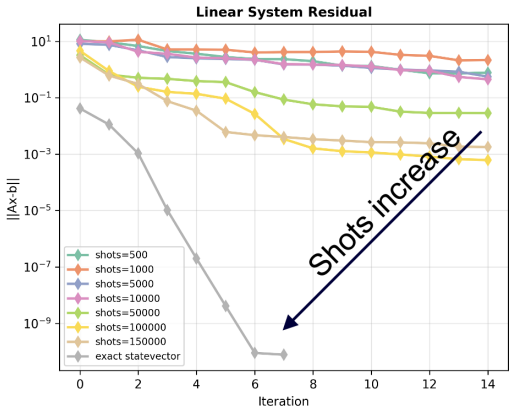


Fig. 9. 1D nozzle: Residual convergence over iterations. Increasing the shots per iteration generally improves convergence and leads to more stable solution tracking across time steps.

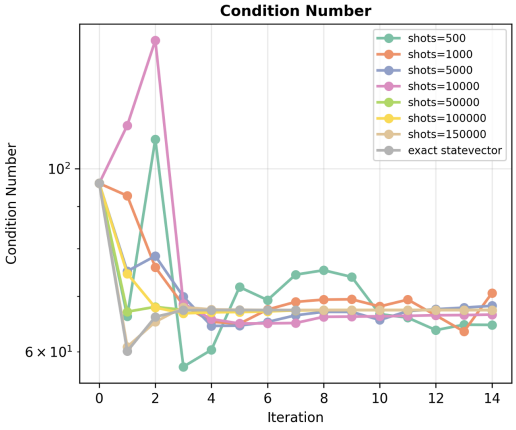


Fig. 10. 1D nozzle: Condition number variation over iterations.

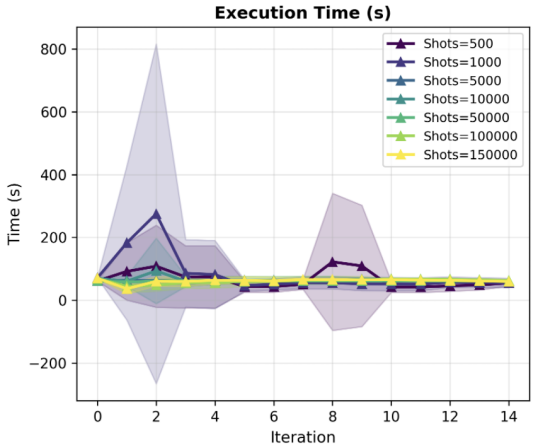


Fig. 11. 1D nozzle: Execution timing analysis. After iteration 4, execution times are almost the same for all shots > 5000. This indicates that the cost of large number of shots is ameliorated after the first few iterations.

and leads to more stable solution tracking across time steps.

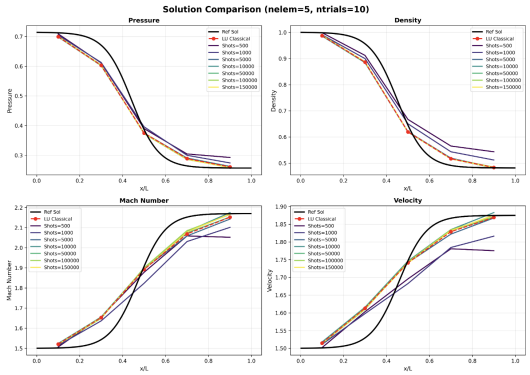


Fig. 12. 1D nozzle: Comparison of runs with increased shots per iteration.

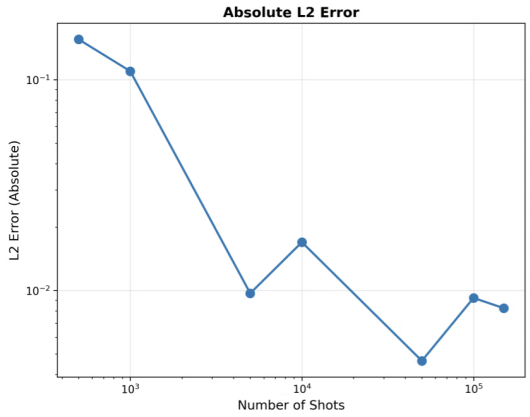


Fig. 13. 1D nozzle: L2 error analysis over iterations for different shot counts.

By repeated runs we can understand the variability present as an intrinsic part of the quantum simulation process and quantify that uncertainty. Here we see that increasing the number of shots reduces the uncertainty, as expected, but there are diminishing returns at higher shot counts.

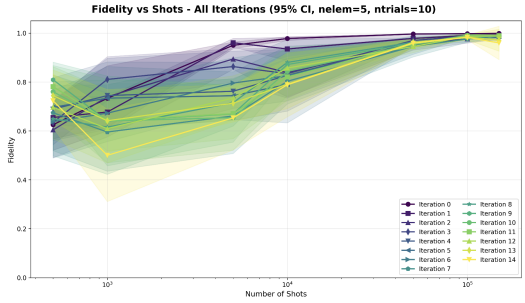


Fig. 14. 1D nozzle: Uncertainty quantification across multiple trials.

3 Key Learnings

3.1 Error and Mitigation in Quantum CFD Circuits

Our study of the HHL algorithm revealed several challenges when applied to CFD problems on current quantum hardware and simulators. Chief among these is the impact of noise and errors on the fidelity of the computed solutions. In this section, we analyze the sources of error in our quantum CFD circuits, their effects on solution accuracy, and potential mitigation strategies.

A heat map of qubit interactions across all circuit layers is shown in Figure 15. This visualization highlights the connectivity patterns and gate applications throughout the circuit, revealing areas of high interaction density that may be more susceptible to error accumulation.

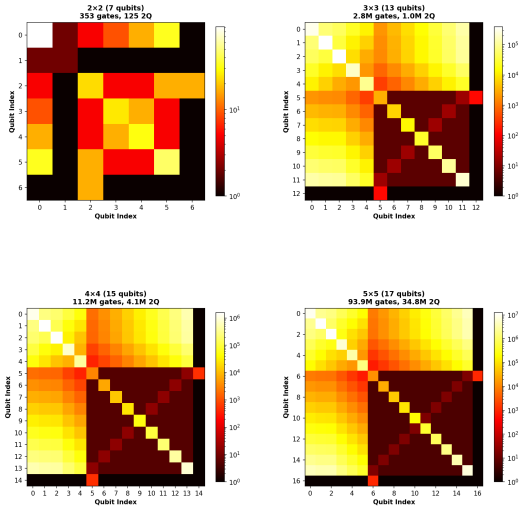


Fig. 15. 2-qubit interaction patterns across all circuit layers

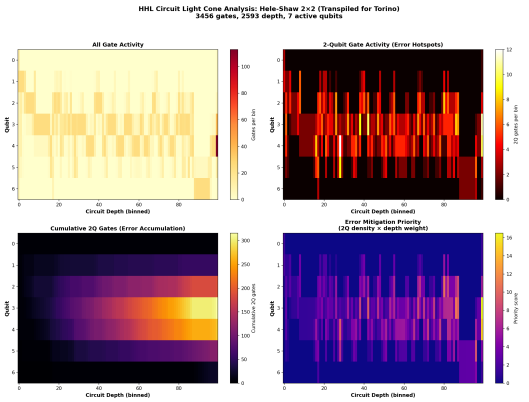


Fig. 16. Qubit activity over circuit depth. Depth requires us to bin the circuit into segments for visualization.

The heat map analysis reveals several key insights about where to focus error mitigation efforts. In the Hele-Shaw (2x2 matrix) case, qubits 2-4 exhibit the highest 2-qubit gate density throughout the circuit, with qubit 3 accumulating approximately 300 total 2Q gates, representing the highest error exposure. The early circuit (bins 0-20) shows intense 2Q activity corresponding to the QPE phase, followed by a mid-circuit spike around bin 25 during eigenvalue rotation, and renewed high activity in the late circuit (bins 60-90) for inverse QPE. For error mitigation priority, the focus should be on qubits 2-4 in the later half of the circuit, with the bright spots at bins 80-95 on qubits 0-4 being highest

priority since errors here have less time to average out before measurement.

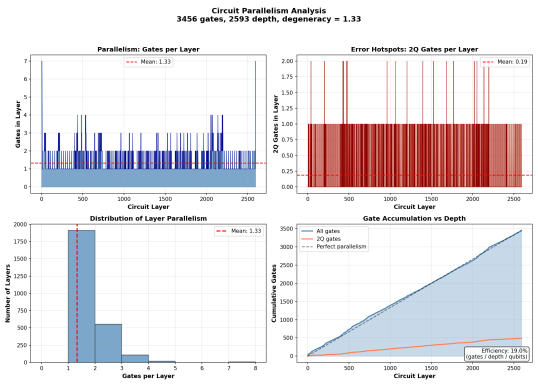


Fig. 17. Circuit parallelism analysis showing gate density per layer

Analysis of circuit degeneracy reveals critical limitations in parallelism. The circuit exhibits a degeneracy of only 1.33 gates per depth layer, indicating very low parallelism, with 73.6% of layers containing only a single gate resulting in mostly sequential execution. While maximum parallelism of 7 gates (all qubits active simultaneously) is theoretically possible, it occurs rarely in practice. The overall efficiency of 19% (gates / depth / qubits) demonstrates substantial idle time across the qubit array. The 2-qubit gate pattern shows clustered bursts with gaps between them, corresponding to controlled operations in QPE. The gap between actual gate accumulation and perfect parallelism shows significant wasted potential, confirming the circuit is highly sequential. This suggests the HHL circuit structure inherently limits parallelism, making depth and thus decoherence time the dominant error source rather than gate count.

In the 1D nozzle case study we have also showed how condition number increases over iterations, and in doing so grows the circuit complexity considerably. Thus practical approaches to preconditioning are essential to keep the circuit depth manageable. We also notice that an increase in qubits improves the precision of the QPE step, however, it also increases the depth of the circuit, which on current quantum hardware can lead to a decrease in overall solution fidelity. Experience in balancing these trade-offs is essential in designing quantum CFD algorithms.

Recent advances in error mitigation techniques, such as zero-noise extrapolation and probabilistic error cancellation, offer potential pathways to improve solution accuracy [13], [14]. However, these methods often require additional circuit executions and classical post-processing, which may not be feasible for large-scale CFD problems. However, new advances in the construction of circuits permits focused error mitigation on the most error-prone qubits and circuit segments, as identified in our heat map analysis. Targeted application of error mitigation techniques to these critical areas may yield significant fidelity improvements without prohibitive overhead. We recommend further research into adaptive error mitigation strategies that leverage circuit structure and qubit error profiles to optimize resource allocation.

In summary, our error analysis highlights the significant challenges posed by noise and decoherence in quantum CFD circuits. While current hardware limitations restrict the practical utility of quantum CFD, targeted error mitigation strategies informed by detailed circuit analysis may help bridge the gap toward useful solutions in the future.

3.2 Quantum CFD Key Learnings

The following are key learnings from the case studies performed for applicability of HHL to CFD:

At this stage, quantum simulation time is much longer than classical solvers. There is no immediate industrial utility based on a performance improvement with the current status of quantum algorithms and hardware at this time. We argue that this does not preclude continued study and value in developing and benchmarking quantum CFD solvers based on fidelity, uncertainty, and cost.

Because of simulation cost, we are limited to small mesh sizes. By small we mean industrially insignificant. The current limits are best suited for research-based demonstrations and other proof of concept exercises.

On idealized hardware, the quantum HHL algorithm is able to solve the Euler equations in 1D accurately. This can be achieved today with noise-free simulators, and in future on error-corrected hardware (see estimations of that projected timeframe in section 4.3). With statistical noise included, the HHL algorithm converges to the true solution but with a very high number of shots/runs.

Matrix preconditioning/or reducing condition number to O(1e3) is essential. A typical CFD matrix is not tridiagonal and without conditioning is not viable for HHL. There are a variety of methods to pre-condition the matrix based on different quantum algorithms, and some of these are candidates for exploring in future work.

Even with preconditioning, the circuits which are created by HHL are prohibitively deep. On simulators, this depth results in very long runtimes. On real quantum hardware, the depth exceeds coherence times and gate fidelities, resulting in unusable results. New techniques to reduce circuit depth are needed.

HHL provides limited information. In it's basic implementation, it does not provide a solution, just a characterization of the solution, for example just the direction of the vector x. One needs to properly scale the result and also add some phase shift algorithm to capture the correct sign of the solution. The case can be reformed into a set of observables, but this requires full state tomography, which is not practical for large problems.

The basic un-enhanced HHL algorithm on real quantum computers in the NISQ era is so noisy as to be unusable, problems so small as to have no practical utility. The circuits generated by HHL are challenging to run in the near-term, being notoriously deep and therefore unable to remain coherent for the duration of the runtime. Running on real quantum hardware or simulation thereof is futile, and running on idealized hardware becomes a test of the HHL method in some distant future where the quantum hardware is near-perfect. This suggests interim approaches (e.g., variational methods which themselves are problematic) should be considered as next steps in this research. However, on clean simulators HHL can show useful fidelity,

and suggests that in future post-NISQ era HHL may yet have utility for CFD. As such, modifications to the HHL algorithm to improve its performance characteristics are an active area of research.

The size of the problem forces a minimum shot count. There is a minimum shot count below which there is no useful information and steps must be made in post-processing to avoid *NaN* and other fatal numeric errors. There must be sufficient shots to show signal over noise.

Dimension does not matter much, what matters is complexity of physics. 1D Nozzle is a nonlinear system of equations (Euler). The 1D Hele-Shaw is a linear uncoupled system of two equations (pressure and velocity) in 2D.

HHL devolves into an Ax=b problem. This can be viewed independently of CFD and benchmarked against existing matrix solvers. Mathematical interest in quantum computing may precede engineering utility.

An increase in qubits does not always translate to an increase in fidelity. Qubit count is not the only measure of quantum computer performance.

Increases in shot counts often improve fidelity and reproducibility (decrease in uncertainty), but to a limit. The HHL progress flattens out as shot count increases.

3.3 Additional Software-centric Learnings

Much was learned from the exercise regarding the applicability of HHL to CFD. Some additional learnings which come from more of a software perspective include:

Simulate with a random seed. Simulators such as Qiskit Aer will produce repeatable results unless a random seed is specified. While repeatable results, such as in estimation values, are an excellent way to debug code, it is not realistic relative to real quantum computers which will have noise and other sources of variation. In uncertainty quantification studies this is particularly important to vary.

Use application sandboxing. Whether it's a container like Docker or a virtual machine, or a simple Python virtual environment, it is important to use a sandbox to keep dependencies separate between different steps in a workflow. This is important for reproducibility, but also for allowing key portions of the workflow to use the latest most performant libraries.

Stay on the tip of the version tree. Quantum computing is a fast-moving field and staying at the top of the version tree is important for performance, keeping with the latest experimental features, etc. This is not yet enterprise software - new versions should be adopted liberally.

LLM software tools are good at porting. The latest software tools with LLM assistance are quite good at porting code. Thus they can be leveraged to stay current with the latest libraries. The LLMs are also good at writing test cases, and in specifying workflows.

Use a workflow tool. Quantum computing, like traditional HPC, is not about running a single job, but orchestrating a workflow of jobs. There are many interim steps and artifacts between the initial problem statement and the final result, and it is important to use a workflow tool to keep track of the control and data flow. Understanding the timing of each step is also important for debugging and optimization, and its preferred to use a tool to handle such details. In addition, real quantum computers are subject to continued calibration - capturing the system state at time of run can also be important for understanding the results.

There are limited opportunities for parallelism in HHL. Circuit construction can be a major bottleneck in HHL. This is a classical procedure and one not generally benefited by GPU acceleration. Since the HHL algorithm calls for a QPE and an inverse QPE, the circuit construction can be parallelized to some extent for a factor of 2 improvement. Each QPE contains some usually modest number of unitary gates, the construction of which can be parallelized to some extent.

In a time series solver, the circuit construction is repeated for each time step. Steps are of course informed by each other and cannot be parallelized.

There is an emerging field of Distributed Quantum Computing, enabled by networked quantum computers, which intends to address this issue by breaking down the problem into smaller sub-problems which can be solved in parallel on connected quantum computers [15]. Domain decomposition and parallelism especially for larger cases, and how this will interact with quantum resources, is an important point for future investigation.

Quantum computer time is expensive. The cost of running cases beyond a few toy examples on real quantum computers is prohibitively high. An hour of runtime can easily cost $1000s on cloud providers such as IBM and AWS [16]. The enterprise will need to come to terms with these costs if we intend to perform simulations on real quantum hardware.

Limited free tier access to older quantum computers is available from vendors like IBM. These are useful for beginners running small toy problems for learning basic concepts, but quickly becomes exhausted running loads even those of limited size discussed in this document.

Simulators, while able to mimic the specific noise and performance characteristics of quantum hardware, only scale to a point, even on HPC. But the cost of running on real hardware is prohibitively high unless there is a strong value proposition.

Publishing results on the basis of simulations only is commonplace in the literature.

$Ax = b$ is just part of solving CFD problems. Computational fluid dynamics involves more than solving linear systems of equations. A complete CFD simulation includes mesh generation, turbulence modeling, boundary condition enforcement, and post-processing. The linear system solve is merely one subroutine within the larger computational workflow, albeit the most important one.

Use performant circuit constructors. Since circuit construction is heavyweight, we can use implementations which are more performant than Qiskit in Python. In recent releases of Qiskit the main routines have been re-written for performance in the Rust compiled language. This is another good reason to use the latest libraries - the newer libraries are usually more performant. NVIDIA also provides alternative circuit construction tools which may be worth exploring.

Use new circuit construction techniques. IBM Qiskit continues to innovate in circuit construction and error mitigation techniques, including those applied in a localized manner [17]. Dynamic circuits are another option for circuit construction, permitting mid-circuit measurement and feedback to reduce circuit depth [18].

The common link between CFD and MAT efforts is the software. The stack, the skills, the circuit math, and sometimes the algorithm or at least aspects of them (e.g. QPE, variational).

3.4 Areas for Future Work

Experientially, there is considerably more to learn, and to keep attuned to the latest developments in the hardware and software. The most important activities are centered around this personal and organizational learning, and include:

- Continue to evaluate algorithms relative to the benchmark framework we will introduce in section 4, to extend when possible from 1D to 2D/3D, and to explore novel techniques for the application of quantum computing to fluid dynamics.
- Comparison of the HHL algorithm with Variational Quantum Linear Solver (VQLS) and its variants [19], new work from U. Kansas [20], Gaitan [21], and formulations of the problem using the Lattice Boltzmann Method [22].

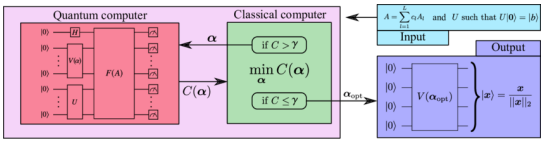


Fig. 18. Variational Quantum Linear Solver (VQLS) approach [19]

- Evaluation of 3rd party vendor solvers some which implement the above mentioned algorithms. This includes for example solvers for reductions to nonlinear ODEs.
- Comparisons of the same CFD algorithms on different hardware types and topologies. It has been anecdotally suggested by ORNL collaborators that n-way qubit connectivity of the type offered by IonQ, Quantinuum, and QuEra, makes a positive impact on these types of results. This would fare poorly for example for vendor machines with less than n-way connectivity such as IBM which uses hex or square lattices.

Some additional areas for future work include:

- Use of the Oak Ridge LuGO approach to the QPE step to reduce circuit complexity [4] and other methods to reduce depth and improve fidelity through error mitigation such as Zero Noise Extrapolation (ZNE) or other techniques applied to all or portions of the solution circuit.
- Exploring use of matrix pre-conditioning, especially those designed for sparse systems and tending to $O(N)$ performance. However, these techniques are classical and often degenerate to $O(N^2)$. This might be useful for a study which does not seek supremacy based on runtime advantage.
- Use of matrix product states and other quantum compression techniques to reduce and simplify the problem size in tolerable ways.
- Modeling the problem as a Hamiltonian and using quantum simulation to solve it, so-called Schrodingerization [23].

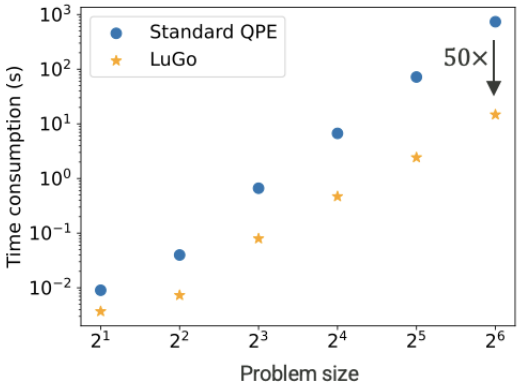


Fig. 19. Oak Ridge LuGO approach reports to reduce QPE circuit complexity

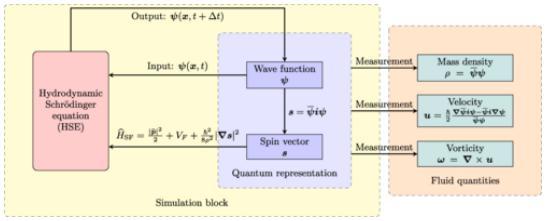


Fig. 20. Schrodingerization approach for quantum simulation [24]

- As recommended by the Ohio Aerospace Institute study [25], consider the molecular-level simulation of the flow problem and therefore its association to quantum chemistry and material science.
- Measure the energy utilization of the quantum solver versus the GPU-accelerated code to assess potential savings. Oak Ridge is interested in this type of study.
- Explore the use of quantum linear solvers as a subroutine in a larger solver.

4 Dynamic Benchmarking for Quantum CFD

4.1 Benchmarking Challenges

To consider a benchmark for quantum CFD, we need to consider what we are evaluating, and to what we are comparing. It would be typical to assess the speedup of an algorithm relative to some prior algorithm on some repeatable problem. It would also be typical to compare the accuracy of the new solver relative to the prior accepted results.

In the case of quantum CFD, we intend to conceed the speed contest to classical solvers now and for the foreseeable future. There is no hardware roadmap which brings sufficient quantum volume to bear in the near-term to be used

to solve the size of industrial problems currently solved by classical CFD entirely on a quantum computer, and therefore there is no basis for conducting a speedup contest at this time. Since sufficient quantum volume cannot be brought to bear, hybrid solutions like VQLS are used, where iterations between the quantum and classical nodes are orchestrated. This is most certainly not faster than a native classical solution due to the transport and other orchestration costs, which are all currently a subject of research interest [26].

For accuracy, noisy near-term quantum computers will not be useful - only when sufficient qubits are available to allocate a good proportion to error mitigation will the remainder of "good" qubits be useful for improving the accuracy of the quantum algorithm. In addition, our classical methods continue to improve, in size and granularity of the problem, accuracy, and speed to solution. We're less interested in how the quantum algorithm is improving over time relative to it's own performance on some fixed reference, and more interested in a comparison of the quantum methods to our current state-of-the-art classical methods, which continue to advance.

The algorithms themselves present challenges to benchmarking. The original HHL is run one time, not iteratively to convergence, as is typical in CFD solvers. Iterative methods are more robust to error, and can be designed to converge despite error in the solution at each step, as long as the error is bounded and does not bias the solution. Quantum linear solvers are being adapted to be used in this manner more consistent with CFD practice, as shown in this report with the 1D nozzle case, and with VQLS, but this is still an area of active research.

Improvements in quantum algorithms may also change the benchmarking landscape. For example, if a quantum matrix conditioning algorithm is developed which can be used as a pre-conditioner to a classical solver, this hybrid approach may be more effective than a pure quantum linear solver approach. The benchmarking framework needs to be flexible enough to accommodate such developments. Improvements in quantum algorithms can lead to improvements in classical, and vice versa, so the benchmarking framework needs to be dynamic and adaptable. The academic community has stood up algorithmic advantage trackers to monitor such fast-moving developments [27], [28].

So in today's terms, quantum CFD fails the comparison with existing classical on all three measures: size, speed, and accuracy. If we conceed speed we can either conceed to classical entirely, or we can benchmark on accuracy and size, which today are so far out of the realm as to be not worthwhile to report. We will instead be interested in a *benchmark we can use to represent line-of-sight to near-term utility*.

4.2 Benchmarking Proposal

We propose a **dynamic benchmarking approach** that monitors evolving quantum CFD methods against our continuously advancing classical state-of-the-art, focusing on accuracy and problem size rather than speed. *When quantum CFD algorithms can address **20% of our problem size** with **80% of our desired accuracy**, industrial utility is in view.*

Here the target problem size is to be selected just-in-time and realistically, e.g. a small open fan, or the tip of a fan, knowing that our largest HPC problems are still often only subsets of total system geometry. In other words, if the current classical problem size is a full blade, quantum may be expected to address 20% of that blade geometry, with 80% of the current classical accuracy.

The benchmark will be re-evaluated periodically, perhaps annually or bi-annually, as both the classical algorithm and the quantum algorithm continue to improve. The frequency of evaluation should match the pace of advancement and proximity to the threshold.

As time goes on, it is expected that the number of algorithms, or variations on algorithms, to be evaluated will increase. Some algorithms may be pure quantum, others hybrid quantum-classical, and there may emerge other classifications. Some will be open source, some already are being commercialized. The use of a hybrid quantum algorithm, or quantum subroutine, may also be used in a future evaluation, for example in plugging in a quantum matrix conditioner prior to running a classical solver. Many variations on the use of quantum algorithms in whole or in part are possible. It will be the task of the research team to select the candidate or candidates for evaluation for any given iteration.

The benchmarking framework should be flexible enough to accommodate these variations. Over time, a suite of benchmark problems may be established, representing different CFD scenarios of interest to our business. These may be grouped into categories by algorithm type, and their relative performance tracked over time relative to each other, the classical state-of-the-art, and the 80/20 threshold. These types of ensemble comparisons are used in other domains, such as computational chemistry and the benchmarking of DFT algorithms and their various implementations [29]. Given the rapidly evolving landscape of quantum computing capabilities and classical CFD advancements, a single static benchmark is insufficient. Instead, we propose an ensemble of benchmarks that can adapt to new developments and provide a comprehensive view of progress in quantum CFD. Over time the list of candidate approaches may grow, or be winnowed down to a smaller set of leading contenders.

We propose an **ensemble of benchmarks** of various quantum CFD algorithms, their various leading implementations, over problem scenarios of interest, continuously updated to reflect the evolving landscape of quantum computing capabilities and classical CFD advancements.

If the 80/20 threshold is reached, actions which might then be taken could also include fitting an existing solver

process with a quantum subroutine and assessing the path of integration into a production solver. It's not recommended that this integration investment be performed until we near that threshold.

4.3 Roadmap to CFD Utility

The purpose of this section is to make a best guess projection of when quantum computing might achieve utility for CFD applications, based on vendor roadmaps and observed progress to date. This projection will inform the proposed benchmarking schedule outlined later in this section.

Qubits do not matter as much as *logical qubits* - error corrected qubits that can be used reliably in algorithms. Current noisy qubits require error correction to be useful for practical applications. The overhead for error correction is high, typically requiring hundreds or thousands of physical qubits to create a single logical qubit, depending on the error rates and the specific error correction code used.

The number of logical qubits required for CFD applications will depend on the complexity of the problem being solved, the size of the system, and the specific quantum algorithms employed. Considering just the size of the mesh in 3D, and a log scaling to qubits, we can envision needing at least 100 relatively error-free qubits, and to achieve this may require as many as 10,000 physical qubits.

We will now look at the vendor roadmaps relative to these needs. We will not attempt to discredit any vendor roadmap, will take them at face value, and will assume that the vendors are motivated to meet their stated goals. We will however down-select which vendors are considered. For this we will be informed by the DARPA Quantum Benchmarking Initiative (QBI), which is conducting a rigorous evaluation of quantum hardware vendors with emphasis on their ability to scale.

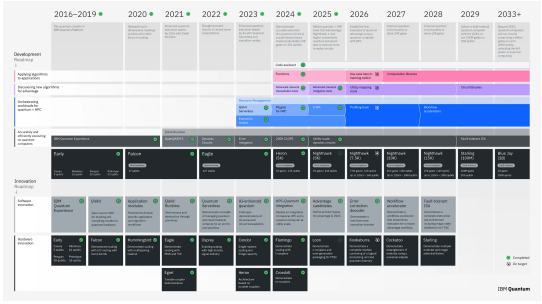


Fig. 21. IBM roadmap, superconducting qubits, lattice connectivity

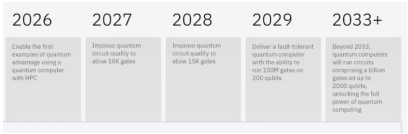


Fig. 22. IBM roadmap detail

We read these roadmaps to indicate that by 2028-2030, several vendors such as IBM and Quantinuum, a spin-off of Honeywell, aim to have systems with thousands to tens of thousands of physical qubits. Assuming error correction

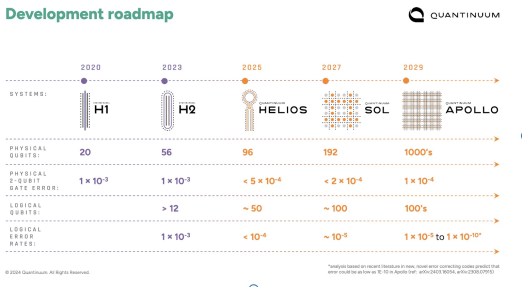


Fig. 23. Quantinuum roadmap, trapped ion qubits, fully connected

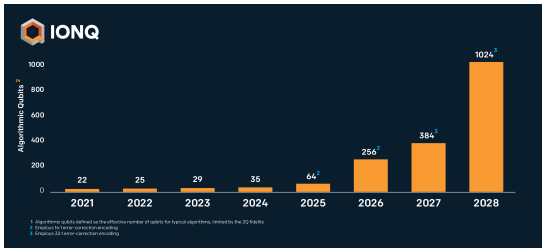


Fig. 24. IonQ roadmap, trapped ion qubits, fully connected

overheads, this could translate to the low hundreds of logical qubits, which may begin to approach the requirements for practical CFD applications. IonQ, a public pure-play quantum company using charged ion qubits, and QuEra a private company which uses neutral atoms, presume to deliver sooner.

4.4 Benchmarking Timeline

Logical qubit counts, connectivity, gate fidelity, and coherence times are all critical factors in determining when quantum computers will be capable of handling CFD workloads effectively. However, useful and performant algorithms which are competitive with classical methods are also essential. Based on the roadmaps presented, we propose a dynamic benchmarking timeline to monitor progress in quantum computing for CFD applications.

TABLE 2
Dynamic CFD benchmarking timeline (proposed)

Year	Task
2025	Establish benchmark framework
2026-27	Annual benchmarking
2028-29	Bi-annual benchmarking
2030	Potential CFD utility?

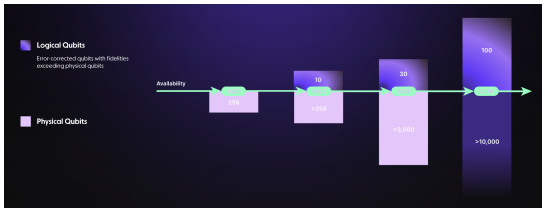


Fig. 25. QuEra roadmap, neutral atom qubits, 2D grid connectivity

In 2026, we will formalize a set of candidate algorithms to evaluate and target their execution on simulators, but also on as many real quantum hardware platforms as possible. This will establish a baseline for performance and identify key bottlenecks. In 2027, we will conduct annual benchmarking to track improvements in hardware and software, adjusting our algorithms as needed to leverage new capabilities.

By 2028-29, as hardware matures, we will shift to biannual benchmarking to capture rapid advancements. Finally, by 2030, we will assess whether quantum computers have reached a level of maturity where they can provide practical utility for CFD applications, based on the accumulated benchmarking data.

4.5 Final Thoughts

The path to quantum utility for CFD is complex and multifaceted, involving advancements in hardware, algorithms, and error mitigation techniques. By establishing a dynamic benchmarking framework and closely monitoring vendor roadmaps, we can position ourselves experientially to be ready to leverage quantum computing as it matures. While challenges remain, the level of investment and innovation in the quantum computing field suggests that practical applications, including CFD, may be within reach within a timeframe requiring increasingly attentive tracking and organizational preparation to adoption.

References

[1] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Physical Review Letters*, vol. 103, no. 15, Oct. 2009. [Online]. Available: http://dx.doi.org/10.1103/PhysRevLett.103.150502

[2] D. Gross, Y.-K. Liu, S. T. Flammia, S. Becker, and J. Eisert, "Quantum state tomography via compressed sensing," *Physical Review Letters*, vol. 105, no. 15, Oct. 2010. [Online]. Available: http://dx.doi.org/10.1103/PhysRevLett.105.150401

[3] S. Jin. (2021) Hhl algorithm. [Online]. Available: https://siyuan-bruce.github.io/2021/11/19/HHL-Algorithm.html

[4] C. Lu, M. G. Meena, and K. C. Gottiparthi, "Lugo: an enhanced quantum phase estimation implementation," 2025. [Online]. Available: https://arxiv.org/abs/2503.15439

[5] Oak Ridge Leadership Computing Facility. (2025) Winter challenge in scientific computing 2025. Oak Ridge National Laboratory. [Online]. Available: https://github.com/olcf/wciscc2025

[6] A. Gallo. (2025) Quantum linear solvers. [Online]. Available: https://github.com/agallojr/quantum_linear_solvers

[7] A. C. Vazquez. (2025) Quantum linear solvers. [Online]. Available: https://github.com/anedumla/quantum_linear_solvers

[8] Oak Ridge Leadership Computing Facility. (2025) Hands-on with frontier. Oak Ridge National Laboratory. [Online]. Available: https://github.com/olcf/hands-on-with-frontier

[9] A. Gallo. (2025) Lightweight workflow manager (lwfm). [Online]. Available: https://github.com/lwfm-proj/lwfm

[10] A. Gallo, I. Claydon, E. Tucker, and R. Arthur, "Industrial experience deploying heterogeneous platforms for use in multi-modal power systems design workflows," in *Accelerating Science and Engineering Discoveries Through Integrated Research Infrastructure for Experiment, Big Data, Modeling and Simulation*, ser. Communications in Computer and Information Science. Springer, Cham, 2022, pp. 257–273. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-23606-8_16

[11] M. Alhawwary, "Fvm euler 1d solver," 2025, gitHub repository, access by permission of the author. [Online]. Available: https://github.com/mhawwary/fvm_euler_1d_solver

[12] M. Gopalakrishnan Meena, K. C. Gottiparthi, J. G. Lietz, A. Georgiadou, and E. A. Coello Pérez, "Solving the hele–shaw flow using the harrow–hassidim–lloyd algorithm on superconducting devices: A study of efficiency and challenges," *Physics of Fluids*, vol. 36, no. 10, Oct. 2024. [Online]. Available: http://dx.doi.org/10.1063/5.0231929

[13] R. Majumdar, P. Rivero, F. Metz, A. Hasan, and D. S. Wang, "Best practices for quantum error mitigation with digital zero-noise extrapolation," 2023. [Online]. Available: https://arxiv.org/abs/2307.05203

[14] R. S. Gupta, E. van den Berg, M. Takita, D. Riste, K. Temme, and A. Kandala, "Probabilistic error cancellation for dynamic quantum circuits," 2023. [Online]. Available: https://arxiv.org/abs/2310.07825

[15] D. Barral, F. J. Cardama, G. Díaz-Camacho, D. Faílde, I. F. Llovo, M. Mussa-Juane, J. Vázquez-Pérez, J. Villasuso, C. Piñeiro, N. Costas, J. C. Pichel, T. F. Pena, and A. Gómez, "Review of distributed quantum computing: From single qpu to high performance quantum computing," *Computer Science Review*, vol. 57, p. 100747, Aug. 2025. [Online]. Available: http://dx.doi.org/10.1016/j.cosrev.2025.100747

[16] Amazon Web Services. (2025) Amazon braket pricing. [Online]. Available: https://aws.amazon.com/braket/pricing/

[17] IBM Quantum. (2025) Samplomatic. [Online]. Available: https://pypi.org/project/samplomatic/

[18] A. C. Vazquez, C. Tornow, D. Riste, S. Woerner, M. Takita, and D. J. Egger, "Combining quantum processors with real-time classical communication," 2025. [Online]. Available: https://arxiv.org/abs/2402.17833

[19] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, and P. J. Coles, "Variational quantum linear solver," *Quantum*, vol. 7, p. 1188, Nov. 2023. [Online]. Available: http://dx.doi.org/10.22331/q-2023-11-22-1188

[20] M. Chaudhary, K. El-Araby, A. Nobel, I. Islam, M. Singh, and S. Ogundele, "A practical quantum solver for multidimensional partial differential equations," in *Proceedings of the SC '25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC Workshops '25. New York, NY, USA: Association for Computing Machinery, Nov. 2025, pp. 1916–1925.

[21] F. Gaitan, "Finding flows of a navier–stokes fluid through quantum computing," *npj Quantum Information*, vol. 6, no. 1, p. 61, 2020. [Online]. Available: https://doi.org/10.1038/s41534-020-00291-0

[22] C. A. Georgescu, M. A. Schalkers, and M. Möller, "qlbm – a quantum lattice boltzmann software framework," *Computer Physics Communications*, vol. 315, p. 109699, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465525002012

[23] S. Jin, N. Liu, and Y. Yu, "Quantum simulation of partial differential equations via schrödingerization," *Phys. Rev. Lett.*, vol. 133, p. 230602, Dec 2024. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.133.230602

[24] Z. Meng and Y. Yang, "Quantum computing of fluid dynamics using the hydrodynamic schrödinger equation," *Phys. Rev. Res.*, vol. 5, p. 033182, Sep 2023. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.5.033182

[25] National Defense Industrial Association, "Ndia drive workshop report," National Defense Industrial Association, Tech. Rep., 2025, manufacturing Division. [Online]. Available: https://www.ndia.org/-/media/sites/ndia/divisions/manufacturing/documents/ndia_drive_workshop-report_v5.pdf

[26] U. Bacher, M. Birmingham, C. D. Carothers, A. Damin, C. D. G. Calaza, A. K. Karnad, S. Mensa, M. Moreau, A. Nober, M. Ohtani, M. Rossmannek, P. Rubin, M. E. Sahin, O. Wallis, A. Shehata, I. Sitdikov, and A. Wennersteen, "Quantum resources in resource management systems," 2025. [Online]. Available: https://arxiv.org/abs/2506.10052

[27] Quantum Advantage Tracker. (2025) Quantum advantage tracker. [Online]. Available: https://quantum-advantage-tracker.github.io/

[28] T. Koch, D. E. B. Neira, Y. Chen, G. Cortiana, D. J. Egger, R. Heese, N. N. Hegade, A. G. Cadavid, R. Huang, T. Itoko, T. Kleinert, P. M. Xavier, N. Mohseni, J. A. Montanez-Barrera, K. Nakano, G. Nannicini, C. O'Meara, J. Pauckert, M. Proissl, A. Ramesh, M. Schicker, N. Shimada, M. Takeori, V. Valls, D. V. Bulck, S. Woerner, and C. Zoufal, "Quantum optimization benchmarking

library - the intractable decathlon," 2025. [Online]. Available: https://arxiv.org/abs/2504.03832

[29] K. Lejaeghere, G. Bihlmayer, T. Björkman, P. Blaha, S. Blügel, V. Blum, D. Caliste, I. Castelli, S. Clark, A. Corso, S. de Gironcoli, T. Deutsch, J. Dewhurst, I. Marco, C. Draxl, M. Dulak, O. Eriksson, J. Flores-Livas, K. Garrity, and S. Cottenier, "Reproducibility in density functional theory calculations of solids," 03 2016.