Technical-report for Project 3: Quantum Algorithm as a PDE Solver for Computational Fluid Dynamics (CFD)

Abdullah Kazi, Hussein Shiri, Tania Jamshaid

Overview: Algorithm Design — Analytic and Numerical Approaches

We implemented two complementary classical methods to establish a baseline:

- **Analytic benchmark:** We derived and used the closed-form travelling viscous shock solution via the Cole–Hopf transform as an exact ground truth.
- **Numerical baseline:** We developed and ran a conservative finite-volume Godunov solver with explicit time stepping and central differencing for diffusion.

Justification of Approaches

- **Cole–Hopf Transform:**
  - Provides an exact, closed-form solution for the viscous Burgers' equation with given initial and boundary conditions and serves as a reliable benchmark for validation.
- **Finite-Volume Godunov Solver:**
  - **Conservation:** The FVM method perfectly preserves the laws of conservation.
  - **Shock handling:** Godunov's method captures shocks by choosing physically correct, entropy-compliant fluxes at discontinuities.
  - **Accuracy:** Central differencing for accurate discretization of diffusion term.
  - **Simplicity:** Explicit time stepping: easy to implement, transparent, stable for CFL constraints.

1. Analytic Benchmark Algorithm (Cole–Hopf Transform)

Steps:

1. **Defined** the viscous Burgers' equation:

$$u_t + u\,u_x = \nu\,u_{xx},$$

with Riemann initial condition

$$u(x,0) = \begin{cases} 1, & x \le 0.5, \\ 0, & x > 0.5, \end{cases}$$

and Dirichlet boundary conditions $u(0,t) = 1$, $u(1,t) = 0$.

2. **Applied** the Cole–Hopf transform:

$$u = -2\nu \frac{\partial}{\partial x} \log \phi,$$

which converted the nonlinear PDE into the linear heat equation

$$\phi_t = \nu \phi_{xx}.$$

3. **Solved** the heat equation for ϕ with the corresponding transformed initial and boundary conditions.
4. **Recovered** the solution $u(x,t)$ from ϕ.
5. **Obtained** the travelling viscous shock solution:

$$u(x,t) = \frac{1}{2} - \frac{1}{2} \tanh\left(\frac{x - x_0 - \frac{1}{2}t}{4\nu}\right),$$

where the shock speed was $c = 1/2$ and thickness $\delta \approx 4\nu$.

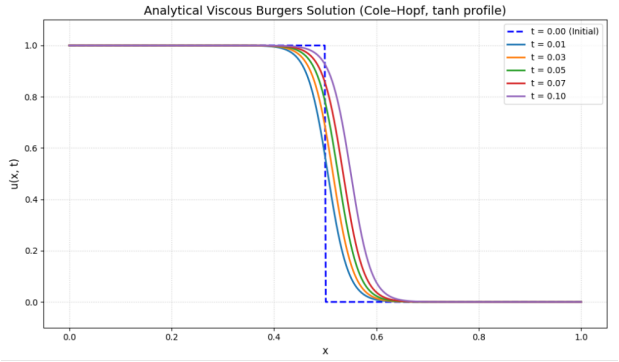

Figure 1: Analytical solution using Cole–Hopf Transform.

2. Numerical Solver Algorithm (Finite-Volume Godunov Method)

Setup: We discretized the domain into $N = 200$ cells with $\Delta x = 0.005$, and set the viscosity $\nu = 0.01$. We selected the timestep as

$$\Delta t = \min\left(0.2\frac{\Delta x}{\max|u|}, 0.5\frac{\Delta x^2}{\nu}\right) = 0.001,$$

with CFL $= 0.2$ and final time $t = 0.1$.

Algorithm steps per timestep:

1. **Computed** interface states u_L and u_R at each cell face.
2. **Evaluated** the Godunov flux $F_{i+1/2}$ for the convex flux $f(u) = \frac{u^2}{2}$ using the scalar Riemann solver logic.
3. **Calculated** the diffusion term with second-order central differences:

$$D_i = \nu \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2}.$$

4. **Updated** the solution explicitly:

$$u_i^{n+1} = u_i^n + \Delta t \left(-\frac{F_{i+1/2} - F_{i-1/2}}{\Delta x} + D_i \right).$$

We recorded snapshots at times $t = \{0.01, 0.03, 0.05, ..\}$.

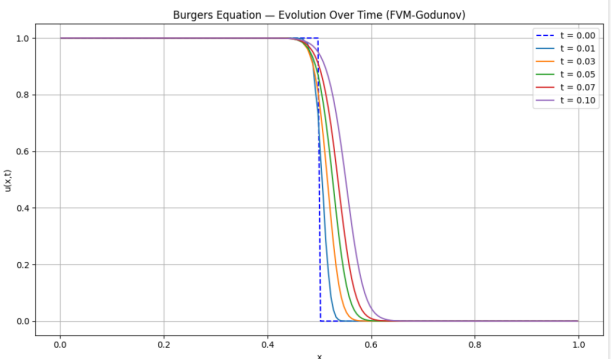

Figure 2: Numerical solution using FVM-Godunov method

1 QTN:

1.1 Methodology:

This method uses a QTN [4] representation of the velocity vector u. In our solution we have implemented two methods, the quantum inspired one as in the paper [4] and a quantum version using trotterization. So we have implemented two solutions for the QTN part of the challenge. The quantum inspired solution uses techniques from the quantum world to implement a classical solution, this solution does not use any gates or circuits and was implemented using the quimb package [2]. While the quantum solution with trotterization uses gates and circuits. The quantum solution includes the following steps:

- The algorithm starts first by a riemann step function of the velocity vector u. $\begin{cases} 1 & \text{if } 0 \leq x \leq 0.5 \\ 0 & \text{if } 0.5 < x \leq 1 \end{cases}$
- u is transformed to psi using the Cole-Hopf transformation, this makes the nonlinear burger's equation linear. $\psi(x,t) = A(t) \cdot \exp\left(\frac{1}{2\nu} \int u(x,t)\, dx\right)$
- The qmprs package [1] is then used to build the MPS, matrix product state, circuit representation of ψ.
- Trotterization [3] is performed on the MPS circuit encoding of ψ.
- The statevector is built. On the noisless simulator it is easily accessed by qiskit built-in functions, while on the noisy simulator and real QPU, the statevector is built from the measurement counts.
- The final statevector represents the evolved ψ vector, we transform it into the evolved u vector by the following formula: $u(x,t) = -2\nu \frac{\partial_x \psi}{\psi}$

Table 1: Results without noise + scaling.

Method	Execution time	L2 error	1 qubit gates u3	2 qubit gates cx	depth	dt	Steps
Godunov fvm	108.772	0.28768	–	–	–	0.001	50
Quantum inspired	0.03	173.90680	–	–	–	0.0001	2000
Trotter 4 qubits	0.014756	0.52750	132	94	160	0.01	1
Trotter 6 qubits	0.387230	3.65813	2774	2178	3417	0.1	4
Trotter 8 qubits	4.759618	8.62814	30358	24362	37115	0.5	10

Table 2: Noisy fake backend results.

	Steps	Depth	1 qubit gates	2 qubit gates	Average T1	Average T2	Average readout error	1 qubit gates error	2 qubit gates error	L2 error	% improvement
No mitigation	1	104	149	50	148s	60.18s	3.73%	0.37183e-3	9.95e-3	0.751	–
	2	203	284	100	148s	60.18s	3.73%	0.37183e-3	9.95e-3	0.955	–
	3	302	419	150	148s	60.18s	3.73%	0.37183e-3	9.95e-3	1.047	–
DD	1	230	465	50	148s	60.18s	3.73%	0.37183e-3	9.95e-3	0.877	no improvement
	2	449	919	100	148s	60.18s	3.73%	0.37183e-3	9.95e-3	1.071	no improvement
	3	668	1373	150	148s	60.18s	3.73%	0.37183e-3	9.95e-3	1.128	no improvement
ZNE	1	n*104	149n	50n	148s	60.18s	3.73%	0.37183e-3	9.95e-3	0.572	23.8178%
	2	n*203	284n	100n	148s	60.18s	3.73%	0.37183e-3	9.95e-3	0.814	14.7%
	3	n*302	419n	150n	148s	60.18s	3.73%	0.37183e-3	9.95e-3	0.927	11.49%

1.2 Noisless results and scaling:

In this section we have compared to the exact/analytical result the classical results, with the quantum inspired one and with trotterization+QTN. It is clear from table 1 that the inspired one gave the worst L2 error. Another important note to mention is that different solvers might reach the same solution at different t. We mean here by "t", the physical t or t in the differential equation and not the time to run on the CPU.

1.2.1 Scaling: grid points

In table 1, the increase in the number of qubits used leads to exponential increase in the grid points. If n is the number of qubits then the grid size is 2^n. It is worth mentioning that we are using here the SuzukiTrotter(order=2) qiskit function. In fact, another function can be used which is LieTrotter() which leads to a circuit with half the depth and half the number of gates approximately but with a little less accuracy. The circuit was transpiled to use the following basis gates ['u3', 'cx'].

1.2.2 Scaling: higher dimensions

The 2D viscous Burgers' equations are:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \quad (1)$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \quad (2)$$

This solution solves the burgers equation in 1D but we can scale it to higher dimensions. First, instead of using the Cole-Hopf transformation we perform carleman linearization to linearize both the linear and the nonlinear operators, then we can solve each of the 2 equations using trotterization.

In fact, before doing so, the 2D velocity vector needs to be transformed into a 1D vector which can be easily done using functions like np.flatten() and the same process can be done for the 3D case but with 3 equations instead of 2. Every equation of these 3 is solved using trotterization. After we solve the different equations we combine the results using techniques like "Quantum Observable Composition" or "Velocity Field Reconstruction".

1.3 Noisy simulator:

In this section we perform the noisy simulation using a Fake backend, specifically FakeManillaV2. The basis gates are ['x', 'cx', 'sx', 'rz']. Due to noise we will use in this section and when running on the real QPU in the next section only the 4 qubits circuit. Since as the number of qubits increase, the depth and gate counts also increase. We have successfully ran the same circuit on the noisy simulator for 3 steps, comparing 2 different error mitigation techniques, dynamic decoupling and ZNE. Table 2 shows clearly that ZNE gave the best decrease in errors approximately 23% while DD provided no improvement for this circuit and the 3 steps. In fact DD gave a little more errors.

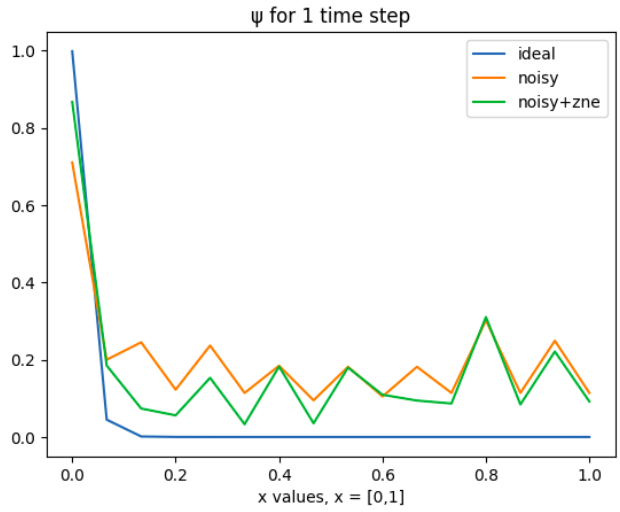

Figure 3: $\psi(x,t) = A(t) \cdot \exp\left(\frac{1}{2\nu}\int u(x,t)\,dx\right)$ on the noisy simulator with and without ZNE error mitigation.

1.4 IQM Garnet QPU:

Now comes the real run on an online QPU from IQM. Iqm-client has a feature that allows running qiskit circuits on their QPU and iqm-benchmarks provides error mitigation features like readout error mitigation and dynamic decoupling. Clearly as in table 3 and for 1 step of the 4 qubits circuit, ZNE again gives the highest reduction in errors around 13% this time instead of 23% with readout error mitigation providing only around 1-1.3% error reduction. The basis gates for this device Garnet are ['r', 'cz'].

1.5 The Hydrodynamic Schrodinger equation and Tensor Network Approach (HSE)

1.5.1 Methodology: Compact HSE+QTN Approach

The Hydrodynamic Simulation Engine (HSE) recasts the 1D Burgers' equation into a quantum framework by mapping the fluid velocity $u(x,t)$ to a two-component quantum wavefunction

$$\psi(x,t) = \begin{pmatrix}\psi_1 \\ \psi_2\end{pmatrix}.$$

The fluid density is recovered as

$$\rho = |\psi_1|^2 + |\psi_2|^2,$$

and the velocity u is derived from the wavefunction's phase and magnitude gradients. The time evolution of ψ is governed by a non-linear Schrödinger-like equation, offering a robust method for handling shocks and sharp gradients.

To manage the exponential growth of the wavefunction's state space, we use **Quantum Tensor Networks (QTNs)**. Wavefunctions are represented as compressed **Matrix Product States (MPS)**, and operators (such as derivatives) as **Matrix Product Operators (MPOs)**. This representation enables efficient and scalable simulations.

The simulation loop proceeds as follows:

1. **Initialization**: The initial velocity profile is converted into dense wavefunctions ψ_1 and ψ_2, which are then compressed into MPS tensors.

2. **Potential Calculation**: At each step, the MPS tensors are used to compute the quantities ρ, u, and the spin-like components s_1, s_2, s_3. These values determine the time-dependent potentials (V_r, V_i, P, Q) required for the HSE evolution.

3. **Time Evolution**: The MPS wavefunctions are evolved by applying an operator constructed from an MPO for the kinetic term and the calculated potentials.

4. **Observable Extraction**: The evolved MPS are converted back to dense arrays to extract the updated velocity field $u(x,t)$, which is then subjected to Dirichlet boundary conditions.

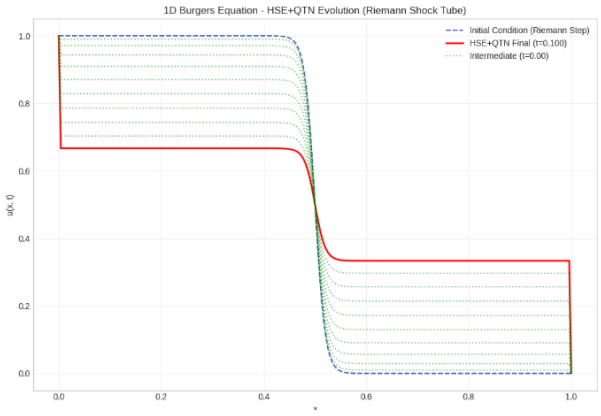

Figure 4: 1
D Burgers Equation - HSE+QTN Evolution (Riemann Shock Tube)

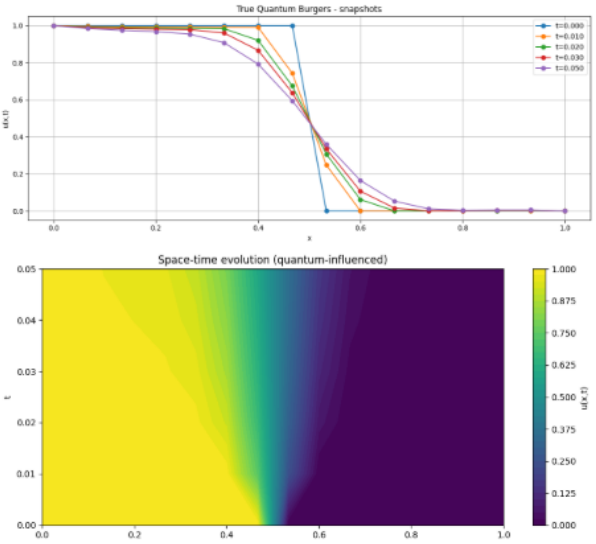

Figure 5: Q
uantum-Classical Burgers' Equation Solver

2 HSE

2.1 Quantum Burgers' Equation Solver

We implement a `TrueQuantumBurgersSolver` class for simulating the 1D Burgers' equation on a quantum computer using Qiskit. The solver initializes physical parameters (viscosity, boundary conditions, grid resolution) and quantum simulation settings (number of qubits, measurement shots, backend). The spatial grid has $2^{n_{\text{qubits}}}$ points and runs on an `AerSimulator`.

The main components are:

- **Initial Condition:** `initial_condition_riemann` sets a Riemann step profile with fixed boundary conditions.
- **Amplitude Encoding:** `amplitude_encode` maps the real velocity field to a normalized complex amplitude vector, adding a phase via the Madelung-inspired cumulative sum.
- **Evolution Circuit:** `build_evolution_circuit` constructs a split-operator style circuit: Quantum Fourier Transform (QFT) for spectral diffusion-like phases, inverse QFT to return to position space, and position-dependent R_Z rotations with controlled-phase couplings to mimic nonlinearity.
- **Execution:** `run_quantum_execution` evolves a pre-prepared statevector through the circuit, samples measurement counts via a multinomial distribution to simulate finite-shot noise, and returns both the sampled counts and a transpiled version of the circuit for backend execution.

This framework allows benchmarking quantum PDE evolution against classical solvers while incorporating realistic quantum measurement noise.

2.2 Hybrid Quantum-Classical Burgers Equation Solver

We also worked on an enhanced hybrid quantum–classical algorithm for solving the 1D viscous Burgers equation using the Madelung transform and Trotterized quantum evolution. The solver initializes a quantum state from a classical step function, encodes the dynamics via separate kinetic and potential evolution circuits (utilizing QFT for kinetic terms and Operator-based phase gates for advection), and alternates quantum time evolution with classical boundary condition enforcement. A trapezoidal phase integration scheme improves numerical stability, while velocity and density fields are extracted classically at each step. The implementation outputs velocity profiles over time and analyzes the depth, size, and gate composition of the final quantum circuit. Numerical experiments demonstrate stable shock formation and highlight the interplay between quantum simulation and classical correction steps.

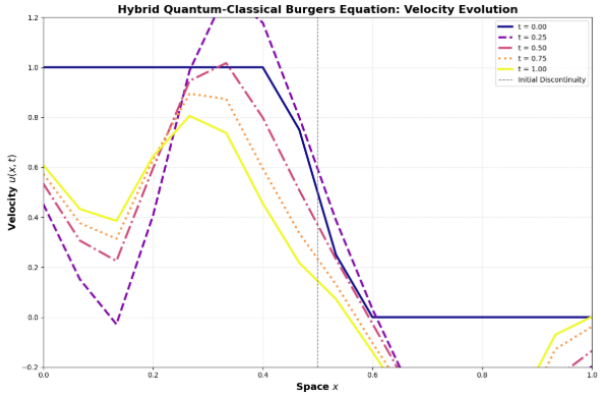

Figure 6: Q
uantum-Classical Burgers' Equation Solver

Table 3: Results on real QPU from IQM, 4 qubits, 1 step, 1000 shots.

Mitigation	(1, 2) qubit gates	Median PRX gate fidelity	Median CZ gate fidelity	Median T1 (μs)	Median T2 (echo) (μs)	L2-error	runtime seconds on QPU	improvement %
Readout error	(90, 50)	99.90%	99.31%	39.89	18.39	0.855	3	1–1.3%
Dynamical decoupling	unknown	99.90%	99.31%	39.89	18.39	0.893	2	no improvement
ZNE Richardson	(90n, 50n)	99.90%	99.31%	39.89	18.39	0.753	8	13.27%
ZNE polynomial	(90n, 50n)	99.90%	99.31%	39.89	18.39	0.768	8	11.63%

References

[1] Tushar Pandey Amir Ali Malekani Nezhad. *qmprs: Quantum Matrix Product Reduced Synthesis*. 2025. DOI: 10.5281/zenodo.15437417. URL: https://doi.org/10.5281/zenodo.15437417.

[2] Johnnie Gray. "quimb: a python library for quantum information and many-body calculations". In: *Journal of Open Source Software* 3.29 (2018), p. 819. DOI: 10.21105/joss.00819.

[3] IBM. *Silicon Quantum Computing announces the world-first integrated circuit manufactured at the atomic scale*. URL: https://qiskit-community.github.io/qiskit-algorithms/tutorials/13_trotterQRTE.html.

[4] R. D. Peddinti et al. "Quantum-inspired framework for computational fluid dynamics". In: *Communications Physics* 7.135 (2024). DOI: 10.1038/s42005-024-01623-8. URL: https://www.nature.com/articles/s42005-024-01623-8.